UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA-CHAMPAIGN

Digitized by the Internet Archive
in 2013

http://archive.org/details/mpixmicroprocess971mcki

1718
UIUCDCS-R-79-971

UILU-ENG 79 1718

MPIX: A MICROPROCESSOR-BASED SYSTEM FOR GRAPHICAL OUTPUT

by

Eric Wallace McKinlay

June, 1979

THE LIBRARY OF THE
OCT 17 1979
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

MPIX: A MICROPROCESSOR-BASED SYSTEM FOR GRAPHICAL OUTPUT

BY

ERIC WALLACE MCKINLAY

A.B., Occidental College, 1977
B.S., California Institute of Technology, 1977

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1979

Urbana, Illinois

TABLE OF CONTENTS

1. Abstract-System Overview 1
1.1 General Description 1
1.2 Design Goals 3

2. Hardware Implementation 5
2.1 Basic System Blocks 5
2.2 Inter-Processor Communication 7
2.3 Input/Output Formats 12
2.4 Character Generation 12
2.5 The Screen Memory and CRT Controller 14
2.6 The Video/RF Interface 17
2.7 Debugging Aids 17

3. Software/Firmware Implementation 21
3.1 Command Processing 21
3.2 Line Generation 22
3.3 Line Clipping 23
3.3.1 Clipping and Drawing Algorithms 27
3.4 Line Transformations 28
3.5 Character Generation 28
3.5.1 Algorithm for Character Generation 29

4. Device Usage 30
4.1 Device Commands 30
4.1.1 Initialization 30
4.1.2 Point Specification 30
4.1.3 Window Definition 31
4.1.4 Transformation Commands 31
4.1.5 Line Drawing Commands 32
4.1.6 Character Commands 33
4.1.7 Region Processing Commands 34
4.1.8 Status Bit Commands 35
4.1.9 Video Adjustment Command 35
4.2 Requirements for Main Processor 36

5. Software Tools 38

List of References 39

Appendices 40
A.1 MPIX Firmware Listing 40
A.2 Main Processor Driver Listing 68
A.3 Command Summary 80
A.4 System Parameters and Timings 82
A.5 Cost Summary 83
A.6 Wirelist for MPIX 84
A.7 Source Listing of the Wirelist Program 115
A.8 Suggestions for Enhancements 123

1. Abstract-System Overview

1.1 General Description

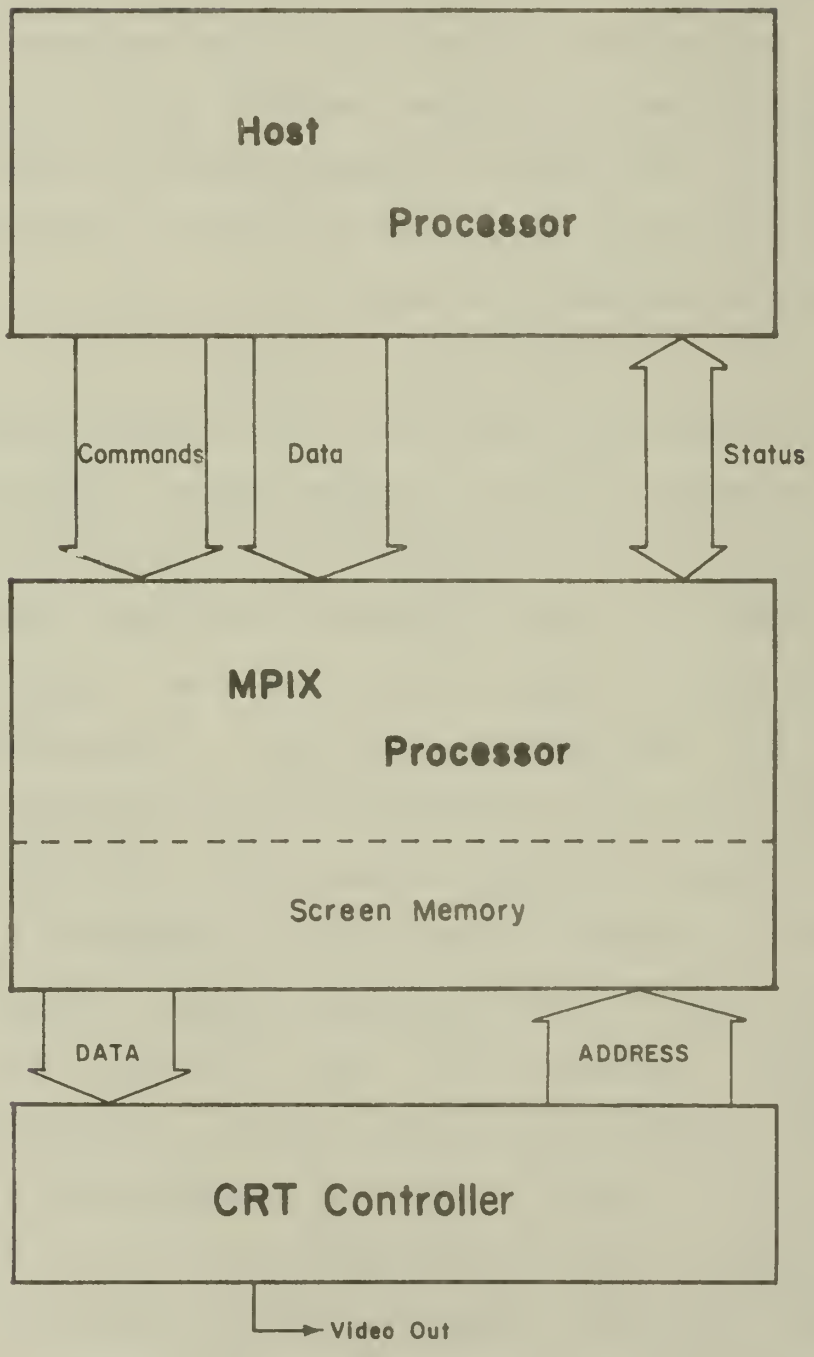
The Micrographix System (MPIX) is designed to enable the microcomputer user to produce medium quality video graphics at modest cost. The graphics produced include simple line drawings and lines of text or combinations of both. The output is produced on a standard television receiver.

The MPIX system operates in parallel with its host computer as much as possible. This allows for maximum utilization of both the host processor and the graphics system. Communication between the host and MPIX is accomplished by way of three standard I/O ports. These ports are accessed in the usual manner by the host computer. Only a small driver need be resident in the main processor to format commands for MPIX. A more complex driver can provide the user with an advanced graphics facility.

MPIX itself contains three basic functional blocks. The arrangement of these blocks is shown in figure 1. The heart of the system is the MPIX processor block. This block controls communication between MPIX and the host and performs all computation needed for graphical output. The processor alters the contents of the display screen by changing the bits stored in the screen memory. Since the contents of the screen memory are constantly displayed, changing a bit has the effect of lightening (or darkening) a particular point on the screen. The scanning of the screen memory is accomplished by the CRT controller section.

Figure 1.

System Block Diagram



This section also generates the the horizontal and vertical synch pulses which are combined with the output of memory to form a composite video signal. This signal may the be sent to a video monitor or, by way of an RF modulator, to a television receiver.

1.2 Design Goals

The following were used as design goals during the formulation of the MPIX system;

1) The MPIX system should be designed to provide a flexible, extensible, and affordable graphics system for secondary school and junior college courses using microcomputers.

2) The system should require a very small ammount of host computer overhead both in terms of time and memory space. To minimize space overhead, all routines which are solely concerned with graphics should be stored and executed by MPIX. To maximize speed, the MPIX system should operate in parallel with the host as much as possible.

3) Attaching the MPIX should require no hardware modifications of the host computer.

4) The MPIX should be flexible enough to allow for future expansion and improvement.

5) The cost of the system should be low enough to allow its purchase by most of the target audience.

6) The system should be extensively documented to promote both learning from and modification of, the system.

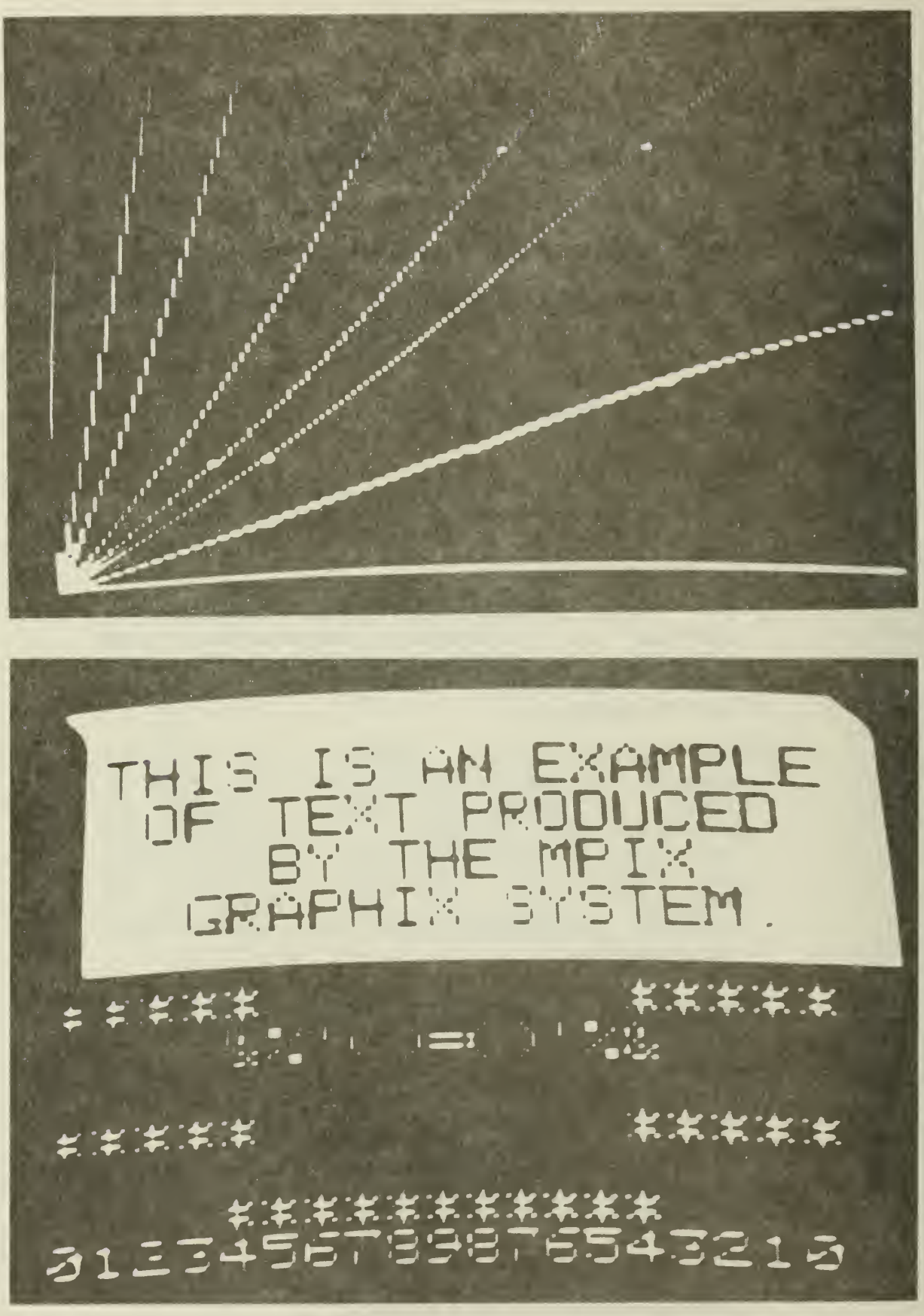


Figure 2. Sample Output from the MPIX.

2. Hardware Implementation

2.1 Basic System Blocks

As shown in figure 3, the system is centered around an Intel 8085 microprocessor CPU. This CPU is connected to Random Access Memory (RAM), Read Only Memory (ROM) and various other devices. The 8085 was chosen largely because of its availability and use of a single, five volt, power supply. Since a cross assembler was available on a large scale computer, this also positively influenced the decision to use the 8085. Unlike many other processors, the 8085 does not have completely separate data and address busses. In order to save pin-outs, the low order address byte is time-multiplexed with the data bus. To enable the processor to be used with standard memory and I/O support chips the low order address byte is latched into the address latch (the 8212 just to the right of the CPU in figure 3) at the beginning of each instruction cycle. The 8155 chip was designed specifically to be used with the 8085 and hence requires no demultiplexing. It provides the system with 256 bytes of RAM, 2 I/O ports, and an interval timer. With the exception of the 8155, all sections of the MPIX communicate with the CPU by way of normal (non-multiplexed) data and address busses.

The ROM is a UV Erasable Intel 2716 which provides the system with 2K bytes of program memory. The X and Y ports are standard Intel 8212's. These ports were used solely to aid in debugging

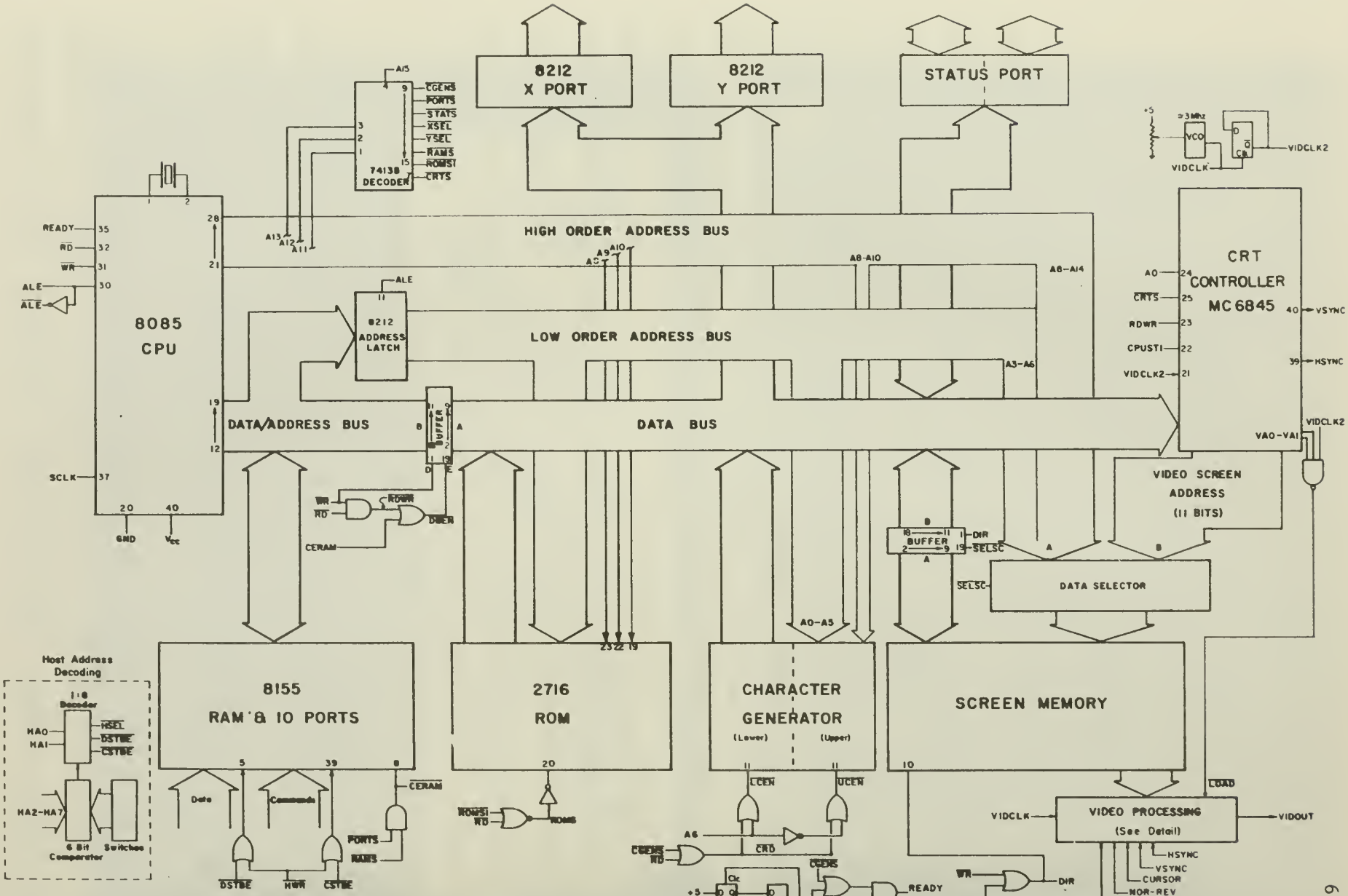


Figure 3. **MPIX System Schematic**

the system and they remain in the design to aid in future enhancements. The most unusual part of the system is the status port. This port is made up of 8 MSI chips and provides bi-directional communication between the processor and the host computer. It also holds status flags used by the software resident in the MPIX. This port is more fully described in section 2.2.

The character generator is simply two 64 character ROMS. The characters within these ROMs are row addressable and measure 6 points by 8 points.

The CRT Controller and Screen Memory work together to generate the video signal as mentioned earlier. The CRT controller is a Motorola MC6845 containing a set of counters and registers allow all of the screen parameters (width, height, etc.) to be changed under software control.

The system memory mapping uses a 1 of 8 decoder attached to A11-A13 to generate the enable signals for the components attached to the data bus. A high level on A15 indicates an access to the screen memory and disables the 1 of 8 decoder. All I/O is memory mapped. Table 1 gives a complete description of the memory mapping used.

2.2 Inter-Processor Communication

Since the host processor and the MPIX are essentially separate processors, we must have some method for coordinating

Table 1

Memory Mapping Locations

Usage	Binary Address	Hex Address	Select Value
ROM	0C00 0000 0000 0000	0000	0
	0C00 0111 1111 1111	07FF	0
RAM	0000 1000 0000 0000	0800	1
	0000 1000 1111 1111	08FF	1
NOT USED	0000 1001 0000 0000	0900	1
	0000 1111 1111 1111	0FFF	1
Y Port	0001 0XXX XXXX XXXX	1000	2
X Port	0001 1XXX XXXX XXXX	1800	3
Status Port	0010 0XXX XXXX XXXX	2000	4
8155 Status	0010 1XXX XXXX X000	2800	5
Command In	0010 1XXX XXXX X001	2801	5
Data In	0010 1XXX XXXX X010	2802	5
Port C	0010 1XXX XXXX X011	2803	5
Timer LOB	0010 1XXX XXXX X100	2804	5
Timer HOB	0010 1XXX XXXX X101	2805	5
Char. Gen.	0011 0000 X000 0000	3000	6
	0011 0111 X111 1111	31FF	6
CRTC Addr.	0011 10XX XXXX XXX0	3800	7
CRTC Data	0011 10XX XXXX XXX1	3801	7
Screen Mem.	1000 0000 X000 0000	8000	-
	1111 1111 X111 1111	FFFF	-

them. One way would be for the host to issue a command every time a task must be performed by MPIX and then simply to wait (i.e. do nothing) until the assigned task is completed. However, since most of the processing done by MPIX is independent of the host, this method would unnecessarily slow the operation of the host. After the host issues a command, it should be free to do other useful work while MPIX performs the given task. Obviously this implies that the host and MPIX should operate in parallel. There must, however, be some communication between the two processors. For example, the host must make sure that a command has been fully received by MPIX before continuing and must not issue a command before the previous command has been completed. To accomplish this coordination, the lower four bits of the the status register are used as a set of 4 binary semaphores. For a complete discussion of parallel synchronization see Shaw[7] chapter 3.

The status register actually contains two four bit registers. One set is used to coordinate with the host processor and one set is used for mode flags by the MPIX. The protocol for the MPIX to host communication is given in table 2. The three semaphores are BUSY,ta referring to the state of MPIX, DATRDY indicating that data is ready to be accepted by MPIX, and DATRCV indicating that data has been received by MPIX. A schematic view of the status port is given in figure 4.

Table 2
Host-MPIX Communication Protocols

State	Data Ready	Data Received	MPIX Busy
Processors Idle	0	0	0
Command Ready	1-H	0	0
Command Received	0-M	1-M	1-M
Host Continuing	0	0-H	1
Data Ready	1-H	0	1
Data Received	0-M	1-M	1
Host Continuing	0	0-H	1
Command Finished	0	0	0-M

M= Change of state due to action of MPIX.
H= Change of state due to action of Host.

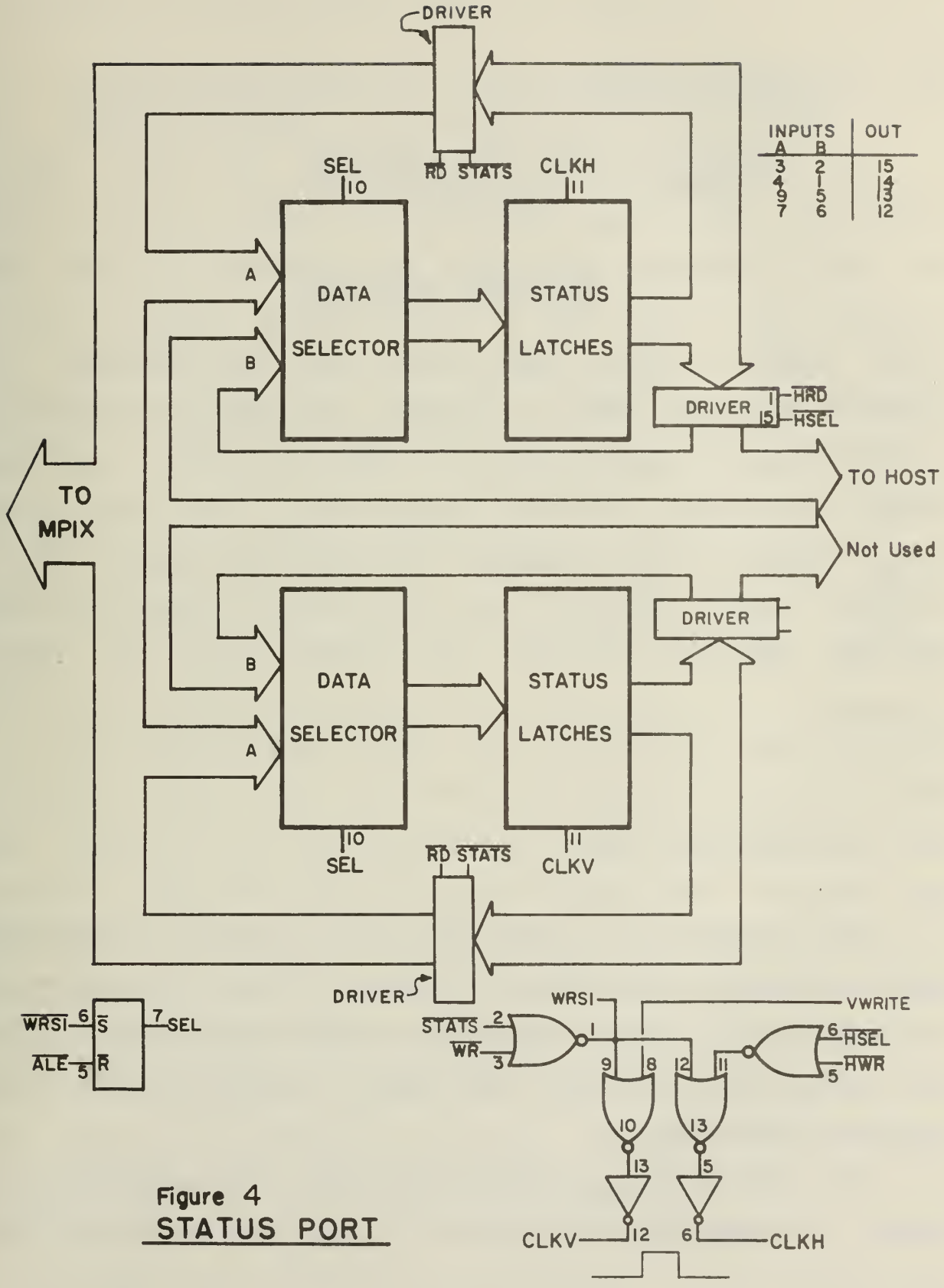


Figure 4
STATUS PORT

2.3 Input/Output Formats

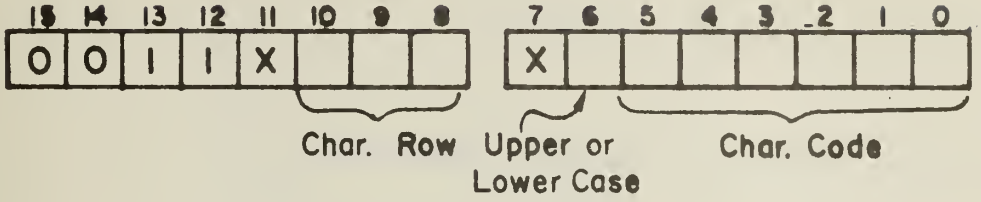
In addition to status information, two other types of information must be sent from the host to MPIX. First, commands are sent to indicate what action is to be taken by MPIX. These commands are in the form of binary numbers between 0 and 127. This provides a great deal of room for future commands to be defined since, at present, only about 30 commands are defined. The second type of information sent to MPIX is data. Each command (with the exception of CHARS) requires from 0 to 4 bytes of data. These bytes are sent immediately following the command itself. A separate port is used for data to avoid any possible confusion between commands and data. The command codes are given in Appendix 3.

2.4 Character Generation

The generation of characters on the MPIX is accomplished either using a ROM standard character set or a user defined RAM character set. The two types may be mixed freely within a string but are treated somewhat differently by the MPIX and will be discussed separately.

The ROM character generator provides 128 characters to the user. This includes both upper and lower case as well as special

characters. The addressing format is as follows;



Character Addressing

Since the ROM is row addressable the row number must be incremented 7 times (for a total of 8 rows) to access a complete character. Each of the rows is 5 bits wide so when a character is fetched only bits 0 through 4 are defined. Because the address format does not match the ASCII character codes a mapping must be performed by the MPIX software. Bit 6 of the address is used to enable one or the other of the two ROMs (for upper and lower case).

The user defined character set is stored in part of the RAM contained in the 8155 chip mentioned earlier. The area between hex address 820 and the bottom of the stack is reserved for this purpose. The stack origin is at hex address 8FA. If twenty bytes are reserved for the stack (providing subroutine and temporary save nesting of up to ten levels) then there remains room enough for 27 user defined characters. These characters are stored in 8 byte blocks beginning at 820. They are stored by rows and only the first 5 bits are used to form the character. For a complete description of the use of user defined characters see sections 3.5 and A.3 of this document. It should be noted that the integrity of the MPIX stack is protected by software. It is therefore not

possible to inadvertently define a character which overwrites the stack.

2.5 The Screen Memory and CRT Controller

All of the information which is displayed on the screen (with the exception of the cursor) is stored in the screen memory which is continually scanned by the CRT controller (CRTC). We will now examine these two sections in some detail. Refer to figure 5 for a schematic representation of these two sections.

The screen memory is made up of 4 2114L static RAM chips with access times of 450 nsecs. These are organized to form a 2K x 8 memory with the upper and lower halves of the screen being stored in separate pairs of chips. This memory must be accessible to both the CPU and the CRTC since it must be both manipulated and displayed. To increase the speed of the MPIX, the CPU is always given access priority over the CRTC. This causes some disturbance of the displayed image when the CPU is accessing the memory. The alternative would be to only allow the CPU to access the memory during vertical and horizontal retrace. This would eliminate the disturbance but would greatly slow the operation of the MPIX.

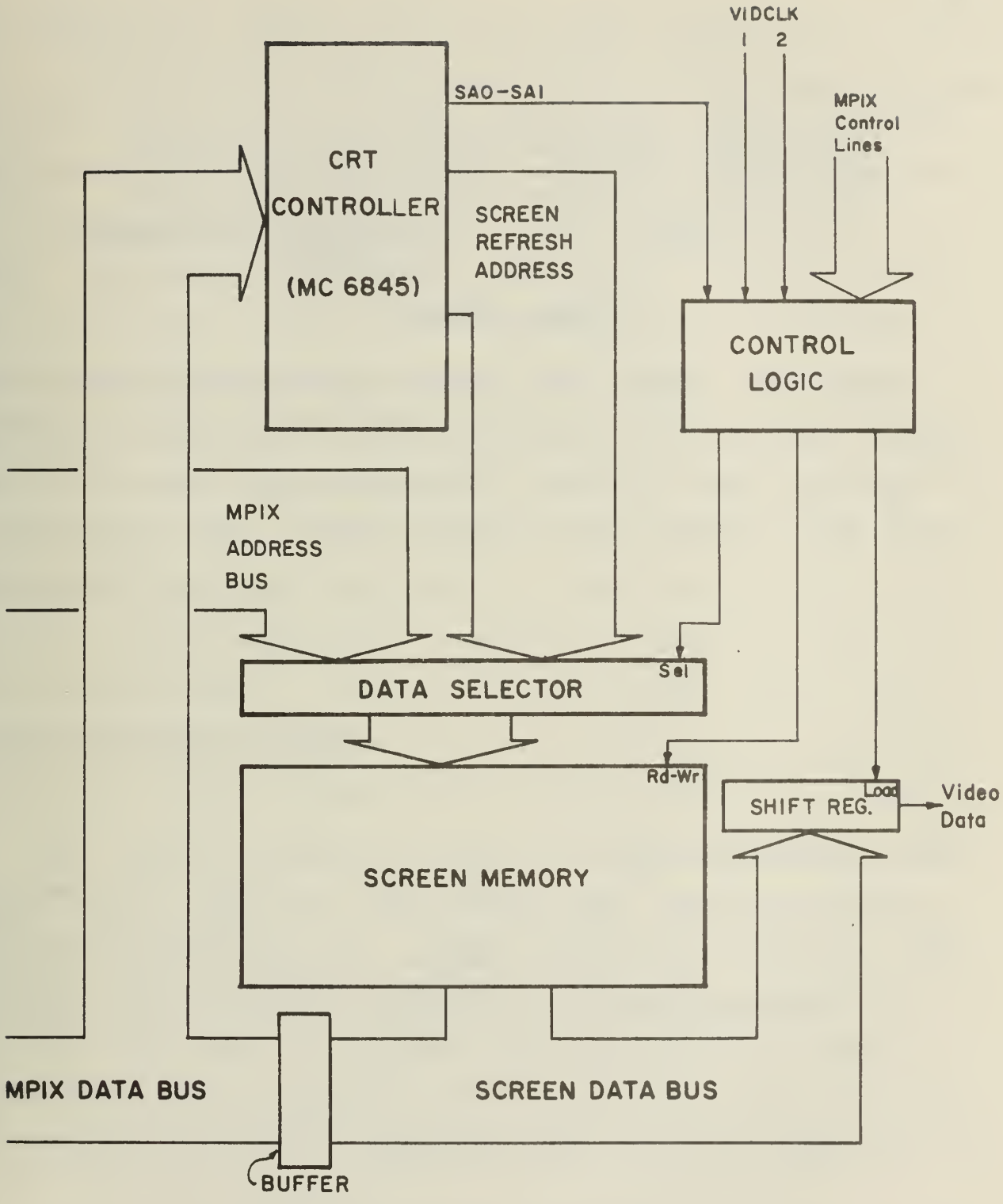
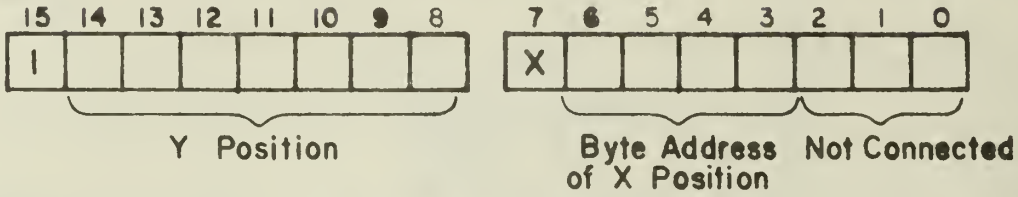


Figure 5 Screen Memory

Access to the screen memory by the CPU is made by standard read and write operations. The addressing format is given below;



Screen Addressing

Generation of the video signal begins with the video clock (VIDCLK) produced by the voltage controlled oscillator (VCO) shown in the upper right corner of the system schematic (figure 3). The frequency of this oscillator is about 3MHz. It is adjustable to allow tailoring the MPIX to individual television receivers. Since the CRTC requires a frequency of less than 2.5MHz, VIDCLK is divided in half before entering the CRTC. This secondary clock (VIDCLK2) then increments all of the counters within the CRTC to provide the address scanning of the screen memory, as well as various other functions.

Eleven address lines (CRTC lines 2 -> 12) are needed to access the 2K bytes of screen memory. The remaining address lines (0 and 1) are combined with VIDCLK2 to produce a load pulse for the video shift register (see figure 5). This load signal is generated every 8 cycles of VIDCLK, yielding a continuous stream of video data out of the shift register.

The horizontal and vertical synch pulses are also generated by the CTRC along with the display and cursor enables. It should be noted that the video data output lags the CRTC address by 8 bit times. This means that when the CRTC address is (0,8), the data being output will be (0,0). Due to this fact it is necessary to "shift" the display enable signal by 8 bit times. This process is performed by the upper flip-flop in the video processing logic (figure 6). The lower flip-flop in figure 6 is used to eliminate a spike in the video signal produced by the parallel load of the shift register.

2.6 The Video/RF Interface

The Video/RF interface is used to convert the composite video signal into a modulated RF carrier which can be fed into the antenna terminals of a standard television receiver. The interface used is the TV-1 manufactured by UHF Associates.

2.7 Debugging Aids

The device shown schematically in figure 7 is used to test and debug the MPIX hardware. It contains a display capable of indicating the values of 16 data lines along with 5 push button

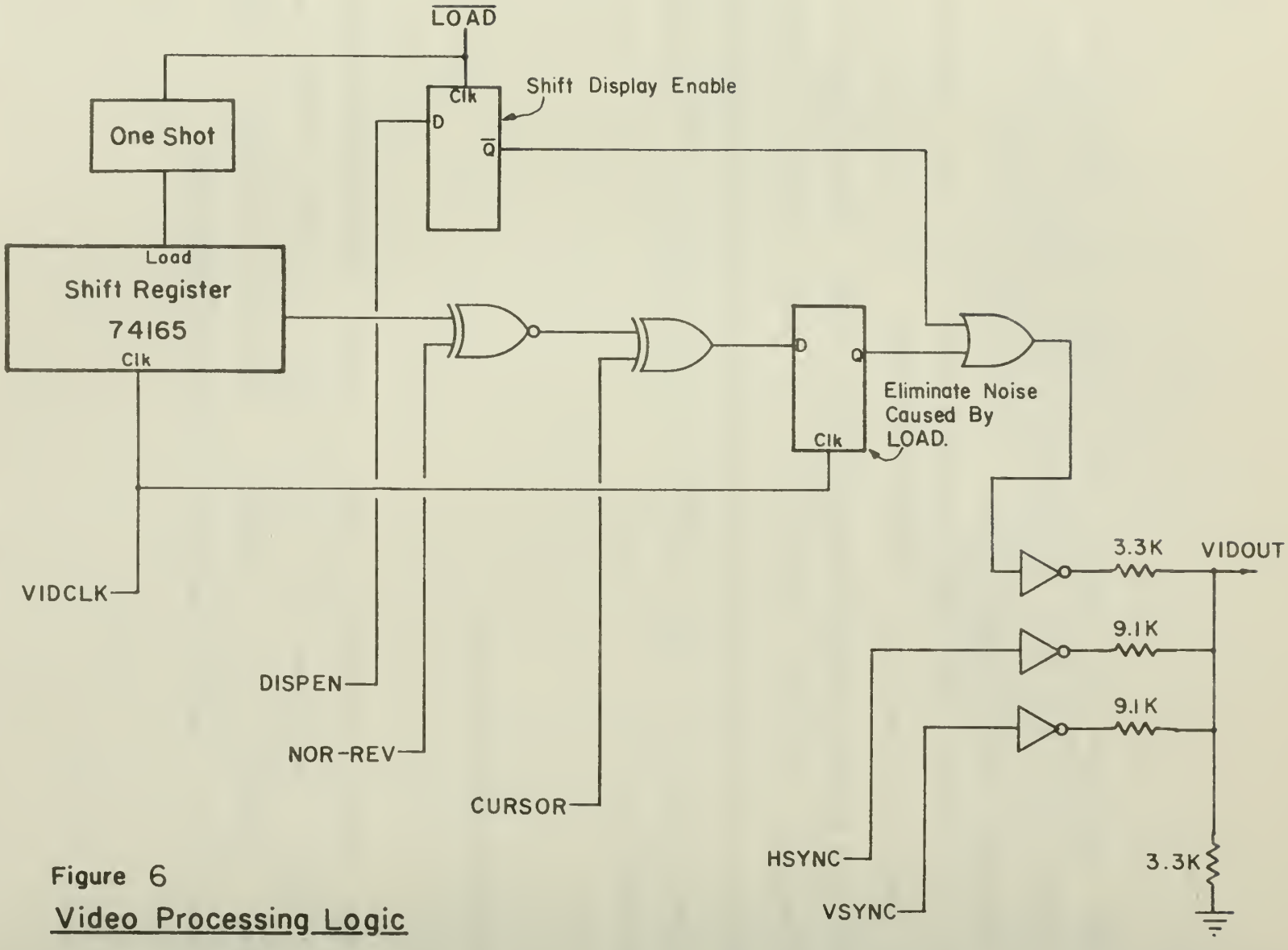


Figure 6

Video Processing Logic

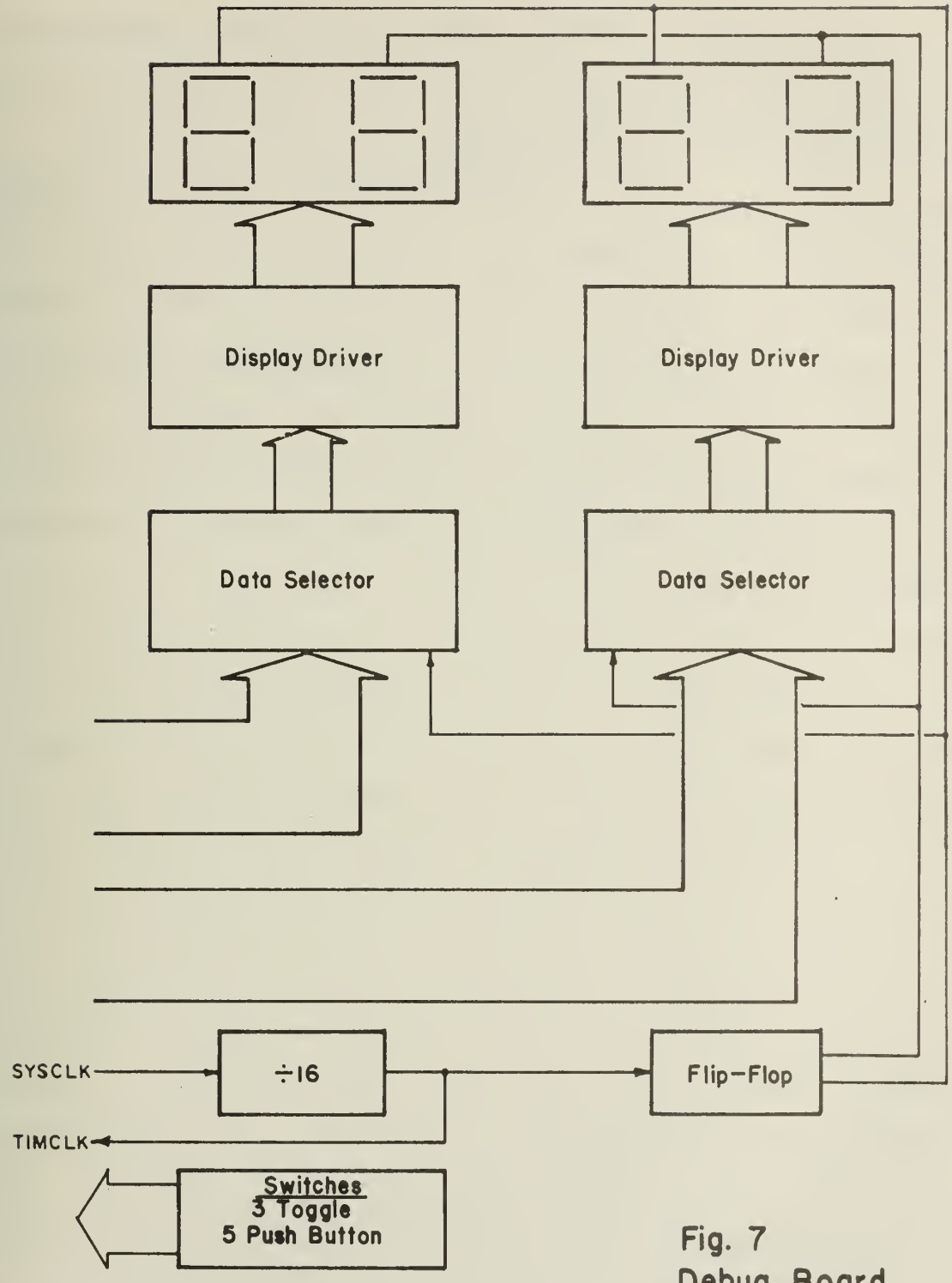


Fig. 7
Debug Board

switches, and 3 toggle switches. The push button switches are used to strobe the input and status ports. The toggle switches supply values for three of the semaphores in the status port.

This device was used to great advantage during the testing and construction of the MPIX. The switches allowed testing to proceed without the presence of a host computer and the display greatly simplified the debugging of the system. The final version of the MPIX firmware includes code which displays ,on the debugger, the contents of the 8155 timer at the completion of each command. This allows commands to be timed within 5 microseconds (the period of the timer clock).

3. Software/Firmware Implementation

The five major tasks performed by MPIX's resident program are described in the following sections.

3.1 Command Processing

The most basic function of MPIX is to receive and interpret commands from the host. This task is performed by the command processor. When MPIX is "idle" it is simply executing a small loop which reads the status port waiting for a data ready signal from the host. When this signal is received, MPIX reads the command, sets the command received and busy flags, and returns to the main body of the MPIX driver. Here the command processor must select one of the command processing routines to be executed. There is one processing routine for each of the possible commands. Transfer to the proper routine is accomplished by making an indirect jump to an address given in a jump table. This table contains the addresses of all the command routines. The displacement into the table is determined by the command received and in turn determines the address to which control is transferred. If data is needed it is requested by the processing routine by making a call to GETDAT. All communication between the firmware and the host is accomplished by the routines GETCOM (get a command) and GETDAT. Regardless of which routine is selected,

when the task is completed the MPIX returns to the idle state awaiting another command.

3.2 Line Generation

The algorithm used for line generation is called the Symmetric Digital Differential Analyzer (DDA). The method is so named because it is derived from the differential equation for a line. Since we are dealing with a discrete approximation to the line, the differential equation is;

$$dX/dY = DELTAX/DELTAY$$

We wish to treat the x and y components of the line separately so we can break this equation into two parts by using a variable N to denote the number of increments between point 1 and point 2. These two equations are;

$$dX/dn = DELTAX/N \quad \text{and} \quad dY/dn = DELTAY/N \ .$$

Note that DELTAX/N indicates the length of each increment along the X axis (and similarly for Y). We must now decide how to choose the value for N and hence the length of the increments. The value chosen cannot be so small that the increment becomes greater than 1 since this would yield a line which contained gaps. The value of N must also not be so large that the resolution of the plotting device causes each point along the line to be displayed a large number of times. Another factor in the choice of N is the fact that division of DELTAX by numbers which are not even powers of 2 is difficult and time consuming. From these

three constraints we want to choose N such that either;

.5 <= DELTAX/N < 1 or .5 <= DELTAY/N < 1 .

These values give the increments which are added iteratively to the starting point (Point 1) of the line drawn. The complete set of equations for the line is then;

```
X(0) = XINITIAL + .5;          Y(0) = YINITIAL + .5;
X(k) = X(k-1) + DELTAX/N;      Y(k) = Y(k-1) + DELTAY/N;
Xdisplayed = Floor(X(k));      Ydisplayed = Floor(Y(k));
```

Using these equations within a loop, points on the line are generated until X(k) = XFINAL and Y(k) = YFINAL (i.e. Point 2 has been reached).

As the points are generated they are placed in the screen memory The "color" (i.e. black or white) of the points is determined by the Field Mode flag contained in the status register. If the field mode is 1 the line will be drawn in white, if the field mode is 0 the line will be drawn in black.

3.3 Line Clipping

The aim of this algorithm is to decide which points, if any, on a given line, fall within a specified window. It can easily be seen that there are basically 3 types of lines relative to the clipping window. These types are:

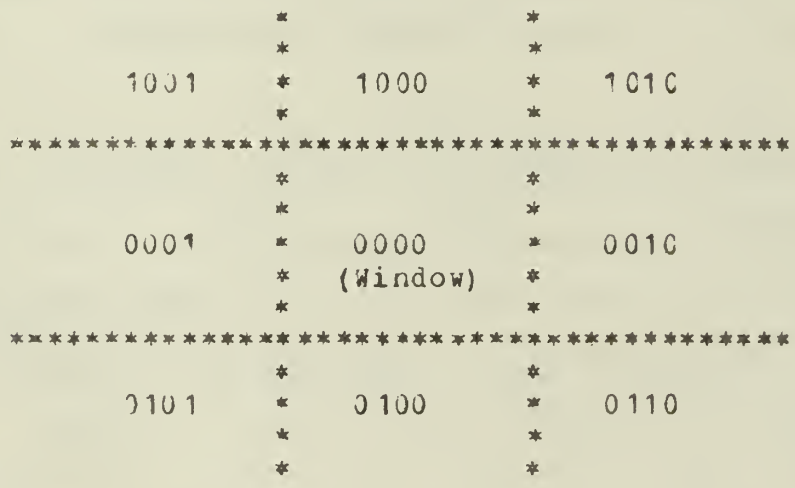
1) Lines which are completely inside the window (completely visible).

2) Lines which are completely outside the window (completely invisible).

3) Lines which lie partially inside the window (partially visible).

Since type 1 and 2 lines require no clipping, it would be much more efficient to detect these types of lines before the clipping process is initiated. The method used is due to Cohen and Sutherland and is described in Sproull [8].

The classification method divides the unclipped picture into nine sectors. Each of these sectors is given a four-bit code. The sectors are labeled in the following manner;

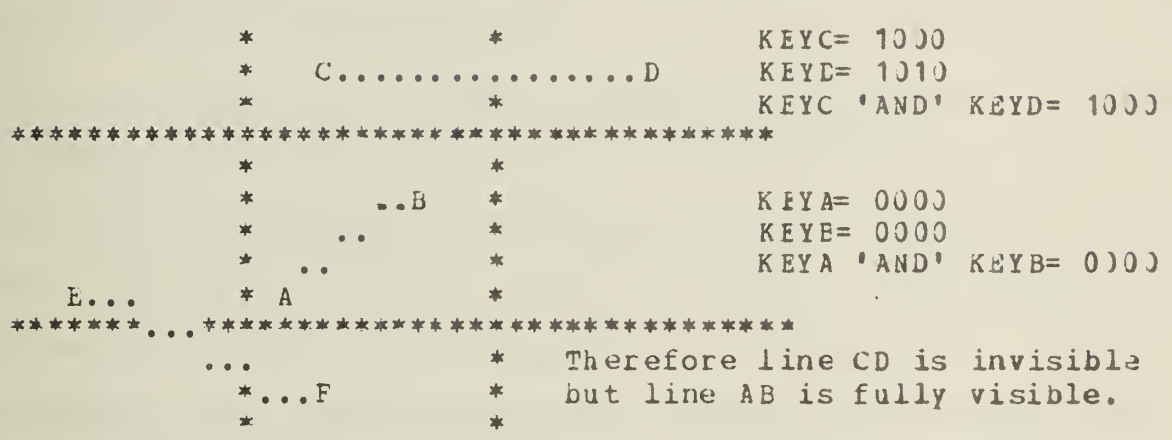


The four-bit code is interpreted so that a bit being set (=1) has the following meaning;

BIT	Meaning
1	The Point is on or above the top edge of the window.
2	The Point is on or below the lower edge of the window.
3	The Point is on or to the right of the right hand edge of the window.
4	The point is on or to the left of the left hand edge of the window.

We can now use these codes to great advantage in performing

the line clipping. Clearly if the four-bit codes (or "KEY"s) for each of the lines endpoints are zero, the line is of type 1 (completely visible). Less obvious is the fact that if the logical "and" of KEY1 and KEY2 is non-zero, the line is of type 2 (completely invisible). Examples of these two types of lines are given below.



Having disposed of the two types of lines which require no clipping, we can now concentrate on type 3 lines, that is those lines which lie partly inside and partly outside the window. Clipping the line requires finding the point nearest to point 1 at which the line intersects the window border. When this is completed, points 1 and 2 are interchanged and the other end of the line is clipped. After both ends are clipped the points are again interchanged and the DRAW routine is called. This insures that the draw routine will never have to be concerned about

whether a line is visible (inside the window) or not.

The method used to find the intersection point is simply a binary search of the line. The program uses a "temporary" point (T) which is moved along the line between points 1 and 2. The direction in which T moves is determined by comparing the key at the temporary point with KEY1. If the 'AND' of the two keys is non zero, T is moved away from point 1. If the 'AND' is equal to zero, T is moved toward Point 1. The magnitude of the distance that T moves along the line is initially set to be one half the total difference between points 1 and 2. Each time T is moved this increment is divided in half. The process terminates when either the proper intersection point is found, or the magnitude of the distance that T moves is equal to zero. After the process terminates it is necessary to check the point which has been found to make sure it lies within the window. This test is necessary because some type 2 lines will not be classified as such by the process discussed earlier (see for example line EF in the preceding figure).

3.3.1 Clipping and Drawing Algorithms

```
PROCEDURE Main
Clip the line.
Reverse points 1 and 2.
Clip the line.
Reverse points 1 and 2.
Draw the line.


PROCEDURE Clip the line.
 Calculate the keys for points 1 and 2 (KEY1 and KEY2).
 IF KEY1 'OR' KEY2 <> 0  (i.e. Line does not cross window).
    Return to calling program.
 ELSEIF KEY1 = 0 OR BORDERKEY1 'EXOR' KEY1 = 0
    Draw the line.
    Return to calling program.
 ENDIF
 Initialize BORDERKEYT to equal BORDERKEY1.
 Calculate DELTAX and DELTAY along with their signs.
 DO FOREVER
    Divide DELTAX and DELTAY by 2.
    IF DELTAX and DELTAY are both 0.
        IF the temporary point (T) is inside the window.
            Draw the line.
        ENDIF
        Return to calling program.
    ENDIF
    Add DELTAX and DELTAY to point1 (the temporary point).
    Calculate the keys for point T (i.e. KEYT).
    Determine the sign of the next X and Y increments.
    IF BORDERKEYT <> 0  (point T is on some border)
        IF BORDERKEYT 'EXOR' KEYT = 0  (T on inside border).
            IF BORDERKEYT 'AND' KEY2 = 0 (T on correct border).
                Draw the line.
                Return to calling program.
            ENDIF
        ENDIF
    ENDIF
 ENDDO


 PROCEDURE Determine the sign of the next X and Y increments.
 IF KEY1 'AND' KEYT = 0
    IF SIGNK = SIGN(DELTAK)                    (make sure that T
        Negate DELTAX, DELTAY and DELTAK        moves toward 1)
    ENDIF
 ELSE
    IF SIGNK <> SIGN(DELTAK)                    (make sure that T
        Negate DELTAX, DELTAY, and DELTAK       moves away from 1)
    ENDIF
 ENDIF
```

```
PROCEDURE Draw the Line
 Calculate DELTAX and DELTAY.
 Output Point 1.
 IF DELTAX and DELTAY are both zero.
    Return.
 ENDIF
 DO until DELTAX or DELTAY contains no leading zeros.
    Shift DELATX and DELTAY to the left.
 ENDDO
 Initialize (x,y) to Point 1.
 Initialize fractional parts of x and y to .5 .
 Increment (x,y) by DELTAX and DELTAY respectively.
 DO FOREVER
    Output the point (x,y).
    IF (x,y) equals Point 2.
        Return.
    ENDIF
    Increment the fractional parts of (x,y) by DELTAX
        and DELTAY.
 ENDDO
```

*****Note: 'OR', 'AND', and 'EXCR' represent bitwise logical operations.

3.4 Transformations

The two endpoints of the current vector are processed according to the current shifting and scaling factors. The shift is performed before the scaling. The scaling is performed by a standard multiplication routine. When the transformation is complete the altered line is stored as the new current vector.

3.5 Character Generation

Characters (both user defined and standard) are placed in the screen memory one row at a time. Upper case characters contain 8

rows and lower case contain 6. Since the character position may be anywhere on the screen, the character row fetched from one of the generators will often need to be shifted before it is placed in the screen memory. If this shift causes the character to cross a byte boundary then two bytes of the screen memory must be fetched and altered for each row of the character.

3.5.1 Algorithm for Character Generation

```
IF output-ascii is a special character THEN
   CASE output-ascii OF
     BS:  Subtract 6 from "X" character position.
     CR:  Begin
             Set "X" character position to zero.
             Process a line feed.
          End.
     LF:  Process a line feed.
     SP:  Add 6 to the "X" character position.
   END CASE.
ELSE
   Map output-ascii into MPIX-char.
   IF end of line THEN
      Set "X" character position to zero.
      Process a line feed.
   END IF
   Calculate the character-row-address.
   IF character is upper case THEN
      ROWS = 8.
   ELSE
      ROWS = 6.
   END IF
   FOR each character row DO
      Shift row into correct position.
      Place character in screen memory.
      Increment character-row-address and screen-position.
   END FOR
   Add 6 to the "X" character position.
   Update the cursor position.
END IF
```

4. Device Usage

4.1. Device Commands

The following sections give a general description of the classes of commands available. Specific command codes and formats are given in appendix 6.3.

4.1.1 Initialization

The INIT command is used to reset the entire system to begin plotting. INIT performs the following functions: 1) Sets the internal stack pointer to a suitable location. 2) Initializes all semaphore flags. 3) Initializes the 8155 including both ports and timer. 4) Sets the current vector (points 1 and 2) to (0.0)->(127.127). 5) Sets the window to be the full screen. 6) Sets the scaling and shifting factors to 1 and 0 respectively. 7) Sets the cursor position to (0.0). 8) Initializes the CRT controller . 9) Clears the screen. Pressing the reset switch on MPIX has the same effect as executing the INIT command.

4.1.2 Point Specification

At all times the endpoints to the "current vector" are held in a save area in the 8155. The point specification commands

alter the endpoints of this "current vector." The user may define either a single coordinate of either end of the vector (requiring 1 data byte after the command), both coordinates of either end of the vector (2 data bytes needed), or both coordinates of both ends of the vector (4 data bytes needed). Point definitions remain in effect until either another point definition is given or the INIT command is executed.

4.1.3 Window Definition

The windowing commands define the area of the screen in which lines will be displayed. Only those lines, or parts of lines, within the window are plotted. This has no effect on lines drawn before the window was specified. The two forms of this command allow the user to define the X (horizontal) window, and the Y (vertical) window independently. The window definitions remain in effect until either another windowing command or the INIT command is executed.

4.1.4 Transformation Commands

The scaling commands allow the user to scale all points sent to MPIT by an integer constant. The scaling of X and Y coordinates are done independently and need not be equal. An expansion is accomplished by a positive scale factor (which

follows the command); a reduction is indicated by a negative scale factor. It should be noted that no checking is performed for overflow of the screen so scale factors should be chosen with care.

Shifting is performed in the same way as scaling. The amount of shift can be either positive or negative. Wrap-around will occur for X and Y values greater than 127.

Shifting and scaling factors are always applied to the current vector when a line drawing command is executed. Since the transformations alter the current vector, their effects on successive vectors are cumulative. As with windowing, transformations have no effect on lines which are already on the screen.

4.1.5 Line Drawing Commands

The execution of the line drawing commands actually places the points along the currently defined line into the screen memory. The command requires no data bytes since the points it uses (the current vector) are already in the MPIX memory. Two forms of the command determine whether the line will be clipped or not. If the line is clipped, the current vector will be altered to the clipped version of the vector. The current transformations are applied to the line just before it is drawn (i.e. after it is clipped).

4.1.6 Character Commands

The Character commands cause either alphanumeric or user defined characters to be placed into screen memory. The dot patterns which form these characters are kept in memory so their generation is quite rapid.

There are 5 commands available for displaying characters. The most basic of these are CHAR and CHARS. These commands output a single character and a string of characters respectively. In either case the only difference between user defined and standard characters is bit 7 of the character code: standard characters are assumed to have a high order bit of 0 while user defined characters have a high order bit of 1. In either case the use of CHAR or CHARS displays the character and updates the current cursor position. The CHARS command continues to accept data bytes from the host until a termination character is received. This character is defaulted to ASCII NULL (00), but may be easily changed.

CHAR and CHARS recognize several special characters. These special characters only affect the cursor position and are the standard ASCII codes for Line Feed, Carriage Return, Back Space and Space. The Carriage Return also performs an automatic line feed.

Each time the cursor position is updated MPIX checks for end of line and end of page. If an end of line is reached an

automatic carriage return is generated. If end of page is reached the screen is scrolled up by one character height.

The CHPO command is used to specify the cursor position. The two data bytes which follow the command give the upper left corner of the desired character position. The HOME command resets the character page and sets the character position to (0,0). No screen clear is done when HOME is executed. The DEFUS command is used to define a new character. Its first data byte gives the character's number (0 -> 27). This number is followed by eight bytes of data which specify the bit patterns for each of the rows of the character. Only the first 5 bits of these bytes are used when the character is printed.

4.1.7 Region Processing Commands

Four commands are available for the processing of regions on the screen. Their effect is to clear or complement a rectangular area of the screen. COMP and COMPR complement the entire screen and a subarea of the screen respectively. COMPR requires 4 data bytes to specify the upper left and lower right corners of the region to be processed. When COMP is executed it also reverses the value of the character mode flag. CLER and CLERR operate in the same way as COMP and COMPR except that the screen, or region of it, is cleared to the current value of the field mode flag.

4.1.8 Status Bit Commands

Two commands are available to manipulate the status flags (bits 4 and 5 of the status port). SCREV complements the field mode flag which complements the screen. Its effect is the same as the COMP command except that it is much faster and the character mode flag is not altered. CHREV reverses the character mode which changes the color with which characters are drawn.

4.1.9 Video Adjustment Command

The CRTAJ command is used to alter the contents of the internal registers of the CRT Controller (CRTC). This provides a method of changing the default parameters for the video display. The default values should be adequate for most cases but use of this command will be necessary for non-standard television receivers and monitors. CRTAJ should be used with care and can only be used after careful study of the CRTC (Motorola MC6845) specifications.

4.2 Requirements for the Main Processor

The minimum software requirements for the host are quite small. Only the routines which directly communicate with the MPIX are absolutely necessary. These routines actually send the commands and data to the MPIX and also must maintain proper synchronization between the processors. The routines given below maintain the synchronization and may be used as the basis for a more complicated driver on the host. Appendix 5.2 gives a listing of a driver which can conveniently be used to test most of the MPIX commands. Both of the routines given below accept their inputs (commands or data) in the accumulator. Only the accumulator is altered by the operation of these routines.

```
STATUS   EQU      70H        ; ADDRESS OF STATUS PORT.
COMPT    EQU      STATUS+2   ; ADDRESS OF COMMAND PORT.
DATPT    EQU      STATUS+1   ; ADDRESS OF DATA PORT.
;
;
;;;;;;;;;; SEND A COMMAND TO THE MPIX ;;;;;;;;;;;
SENDCOM: OUT      COMPT
         IN       STATUS
         ORI      01H        ; SET DATA READY FLAG.
         OUT      STATUS
CWAIT:   IN       STATUS     ; LOOP UNTIL DATA RECEIVED FLAG IS SET.
         ANI      02H
         JZ       CWAIT
         IN       STATUS
         ANI      0FDH       ; CLEAR DATA RECEIVED FLAG.
         OUT      STATUS
         RET
;
;;;;;;;;;; SEND DATA TO THE MPIX ;;;;;;;;;;;
SENDAT:  OUT      DATPT
         IN       STATUS
         ORI      01H        ; SET DATA READY FLAG.
         OUT      STATUS
DWAIT:   IN       STATUS     ; LOOP UNTIL DATA RECEIVED FLAG IS SET.
         ANI      02H
         JZ       DWAIT
         IN       STATUS
         ANI      0FDH       ; CLEAR DATA RECEIVED FLAG.
         OUT      STATUS
         RET
```

5. Software Tools

Several software tools were used for the construction of MPIX. The software for MPIX was developed using the 8080 cross assembler (an Intel product) on the Cyber 175. The 8080 simulator on the Cyber was also used for some preliminary debugging although it proved too slow to be very useful. After the code was generated on the Cyber it was downloaded to a North Star Horizon which contained a BCM programmer. This programmer was then used to transfer the MPIX software to a 2716. The Horizon also was the host processor for the MPIX during final testing. Due to the existence of the debug board (see section 2.7), the vast majority of debugging was done with no host attached.

The Wirelist program (see A.5 and A.6) was an extremely useful tool for keeping track of the hundreds of wires in the MPIX. This program should be maintained and improved since it provides a very useful tool for hardware design. A possible addition is a plotter driver which could provide the user with a component layout (see figure 8).

List of References

1. Intel Corporation. Interp/80 Users Manual. Intel Publication number 98-037A. Intel Corporation. 1975.

2. Intel Corporation. MCS-85 Users Manual (Preliminary). Intel Publication number 98-366C. Intel Corporation. 1977.

3. Corporation. 8080 Assembly Language Programming Manual. Intel Publication number 98-004C. Intel Corporation. 1976.

4. Jones, Vincent C. Graphics Software Standard for Microcomputers. Unpublished Paper.

5. McKinlay, Eric. Documentation for the Line Generation and Clipping System (LCGS). Unpublished Paper. 1978.

6. Motorola Incorporated. MC6845 CRT Controler Data Booklet (Advance Information). Publication AD1-465. Motorola 1977.

7. Shaw, Alan C. The Logical Design of Operating Systems. Prentice-Hall 1974.

8. Sproull, Robert and Newman, William. Principles of Interactive Computer Graphics. McGraw-Hill 1973.

Appendices

A.1 MPIX Firmware Listing

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                               ;
;                        MACRO DEFINITIONS                                      ;
;                                                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;;;;;;;;;;;;;;;;;; LOAD REGISTER FROM 8155 ;;;;;;;;;;;;;;;;;;;;;;;;
;
LRDISP  MACRO     R1,DISP
        LXI       H,DISP
        MOV       R1,M
        ENDM
;
;
;;;;;;;;;;;;;;;;; STORE REGISTER IN 8155 ;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SRDISP  MACRO     R1,DISP
        LXI       H,DISP
        MOV       M,R1
        ENDM
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SHLF    MACRO     DUMY
        RLC
        ANI       0FEH
        ENDM
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
RFOUR   MACRO     DUMY
        RRC
        RRC
        RRC
        RRC
        ENDM
```

```
;;;
```

```
CSTAT     EQU       2800H
TIMEL     EQU       2804H
TIMEH     EQU       2805H
CRTD      EQU       3801H
CRTA      EQU       3800H
SCREN     EQU       8000H
;
;;;;;; HANDSHAKE MASKS ;;;;;;
DATRCV    EQU       02H
DATRDY    EQU       01H
NRDY      EQU       0FEH
BUSY      EQU       04H
NBUSY     EQU       0FEH
PGFUL     EQU       40H
FLMOD     EQU       10H
CHMOD     EQU       20H
;
;;;;;;;;;;;; VARIOUS CONSTANTS ;;;;;;;;;;;
I8155     EQU       48H
ECLN      EQU       75H
CHARH     EQU       30H
INITFL    EQU       34H
NCCMS     EQU       1BH
STIIM     EQU       I8155 OR 0C0H
SPTIM     EQU       I8155 OR 40H
DUMMY     EQU       0
```

```
;;;
```

```
;;;
```

```
        LDA     TIMEL       ; DISPLAY TIMER.
        CMA
        STA     YPORT
        LDA     TIMEH
        CMA
        ANI     3FH         ; CLEAR TIMER MODE BITS.
        STA     YPORT
        JMP     PROCOM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  THE FOLLOWING ARE THE ROUTINES FOR EACH OF THE COMMANDS   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
DEFX1:  CALL    GETDAT      ; SPECIFY THE X COORD OF POINT 1.
        STA     X1
        RET
;
DEFX2:  CALL    GETDAT      ; SPECIFY THE X COORD OF POINT 2.
        STA     X2
        RET
;
DEFY1:  CALL    GETDAT      ; SPECIFY THE Y COORD OF POINT 1.
        STA     Y1
        RET
;
DEFY2:  CALL    GETDAT      ; SPECIFY THE Y COORD OF POINT 2.
        STA     Y2
        RET
;
DEFPT1: CALL    GETDAT      ; SPECIFY BOTH COORDS OF POINT 1.
        STA     X1
        CALL    GETDAT
        STA     Y1
        RET
;
DEFPT2: CALL    GETDAT      ; SPECIFY BOTH COORDS OF POINT 2.
        STA     X2
        CALL    GETDAT
        STA     Y2
        RET
;
DEFVEC: CALL    GETDAT      ; SPECIFY ENDPOINTS OF VECTOR.
        STA     X1
        CALL    GETDAT
        STA     Y1
        CALL    GETDAT
        STA     X2
        CALL    GETDAT
        STA     Y2
        RET
;
WINDX:  CALL    GETDAT      ; SPECIFY THE X LIMITS OF THE WINDOW.
        STA     LXLIM
        CALL    GETDAT
```

```
        STA     RXLIM
        RET
;
WINDY:  CALL    GETDAT      ; SPECIFY THE Y LIMITS OF THE WINDOW.
        STA     BYLIM
        CALL    GETDAT
        STA     TYLIM
        RET
;
DRAWCL: CALL    DCLIP
        CALL    EXCH12
        CALL    DCLIP
        CALL    EXCH12
        CALL    CKEY12      ; RECALCULATE ENDPOINT KEYS.
        LDA     KEY1
        MOV     B,A
        LDA     KEY2
        ANA     B           ; CHECK FOR LINE OUTSIDE WINDOW.
        RNZ
        CALL    DRAWL
        RET
;
SCALX:  CALL    GETDAT      ; SET X SCALING.
        STA     XSCA
        RET
;
SCALY:  CALL    GETDAT      ; SET Y SCALING.
        STA     YSCA
        RET
;
SHFTX:  CALL    GETDAT      ; SET X SHIFT.
        STA     XSHF
        RET
;
SHFTY:  CALL    GETDAT      ; SET Y SHIFT.
        STA     YSHF
        RET
;
CHAR:   CALL    GETDAT      ; GET CHARACTER TO BE PLOTTED.
        CALL    CHARO       ; OUTPUT CHARACTER.
        RET
;
CHARS:  CALL    GETDAT      ; OUTPUT A STRING OF CHARACTERS.
        MOV     B,A
        LDA     STERM       ; GET TERMINATOR CHARACTER.
        CMP     B           ; CHECK FOR END OF STRING.
        JZ      DCHS
        MOV     A,B
        CALL    CHARO
        JMP     CHARS
DCHS:   RET
```

```
CHREV:  MVI     B,CHMOD  ; COMPLEMENT CHARACTER MODE.
        LDA     STATUS
        XRA     B
        STA     STATUS
        RET
;
;
DEFUS:  CALL    GETDAT   ; DEFINE A USER CHARACTER.
        ANI     1FH      ; GET USER CHARACTER NUMBER.
        RLC
        RLC
        RLC              ; GET DISPLACEMENT INTO USER CH TABLE.
        MOV     E,A
        MVI     D,00H
        LXI     H,UCHB   ; GET LOWEST USER CHAR ADDRESS.
        DAD     D        ; ADD DISPLACEMENT TO BASE ADDRESS.
        XCHG
        LXI     H,00H    ; CLEAR HL REGISTER.
        DAD     SP       ; GET CURRENT VALUE OF STACK POINTER.
        MOV     A,L
        SUI     08H      ; HL <- MAX BEGIN CHAR ADDRESS.
        MOV     L,A
        MOV     A,H
        CMP     D        ; INSURE THAT CHAR WILL NOT OVERLAP
        JC      OVFLS    ;                 THE STACK.
        MOV     A,L
        CMP     E
        JC      OVFLS
        JMP     NOVFL
OVFLS:  LXI     D,UCHB   ; ON ERROR PUT NEXT CHAR IN BASE ADDR.
NOVFL:  XCHG             ; HL <- CORRECTED BEGIN ADDRESS IN RAM.
        MVI     A,08H
        STA     LPCN1    ; INITIALIZE LOOP COUNTER.
DULP1:  CALL    GETDA1   ; GET NEXT BYTE OF CHARACTER.
        ANI     3FH
        MOV     M,A
        INR     L        ; MOVE TO NEXT RAM POSITION.
        LDA     LPCN1
        DCR     A
        STA     LPCN1
        JNZ     DULP1
        RET
;
;
SCREV:  LDA     STATUS
        XRI     FLMOD
        STA     STATUS
        RET
;
;
HOME:   LXI     H,8000H  ; SET CURSOR AND START ADDRESS TO (0,0).
        SHLD    CHPOS    ; CHAR POS <- (0,0)
        MVI     A,00H
```

```
        MOV     H,A
        CALL    PUTCUR
        MVI     A,PGFUL
        CMA
        MOV     B,A
        LDA     STATUS
        ANA     B           ; SET PAGE FLAG TO NOT FULL.
        STA     STATUS
        MVI     A,STARH
        STA     CRTA
        XRA     A
        STA     CRTD        ; SET START ADDRESS (H) TO 0.
        STA     SADDH
        MVI     A,STARL
        STA     CRTA
        XRA     A
        STA     CRTD        ; SET START ADDRESS (L) TO 0.
        STA     SADD
        RET
;
;
CHPO:   CALL    GETDAT      ; DEFINE THE CHARACTER POSITION.
        MOV     L,A         ; L <- X POSITION.
        CALL    GETDAT
        ORI     80H         ; INSURE THAT CHPOS IS ACTUAL ADDR.
        MOV     H,A         ; H <- Y POSITION.
        SHLD    CHPOS
        CALL    PUTCUR
        RET
;
;
CLER:   LXI     D,7F7FH     ; CLEAR THE ENTIRE SCREEN.
        LXI     H,0000H
        MVI     A,0FFH
        STA     CLCMP
        CALL    PROREG
        RET
;
;
CLERR:  CALL    GETDAT      ; GET UPPER LEFT CORNER (X FIRST).
        MOV     L,A
        CALL    GETDAT
        MOV     H,A
        CALL    GETDAT      ; GET LOWER RIGHT CORNER (X FIRST).
        MOV     E,A
        CALL    GETDAT
        MOV     D,A
        MVI     A,0FFH
        STA     CLCMP
        CALL    PROREG
        RET
```

```
COMP:   LXI     D,7F7FH   ; COMPLEMENT THE ENTIRE SCREEN.
        LXI     H,0000H
        MVI     A,00H
        STA     CLCMP
        CALL    PPOREG
        LDA     STATUS
        XRI     CHMCD
        STA     STATUS
        RET
;
;
COMPR:  CALL    GETDAT    ; GET UPPER LEFT CORNER (X FIRST).
        MOV     L,A
        CALL    GETDAT
        MOV     H,A
        CALL    GETDAT    ; GET LOWER RIGHT CORNER (X FIRST).
        MOV     L,A
        CALL    GETDAT
        MOV     D,A
        MVI     A,00H
        STA     CLCMP
        CALL    PPOREG
        RET
;
;
;
;;;
```

```
;;;
```

```
        CALL    CSIGN       ; CALC SIGN OF THE TWO DELTAS.
        LDA     KEYT        ; TEST FOR BOUNDRY INTERSECTION.
        ANI     0F0H
        JZ      CLIP
        LRDISP  D,KEYT
        RFOUR   DUMMY
        XRA     D           ; TEST FOR INTERSEC WITH INSIDE BNDR.
        ANI     0FH
        JNZ     CLIP
        LDA     KEY2        ; TEST FOR INTER W/CORRECT INSIDE BNDR
        RFOUR   DUMMY
        ANA     D
        ANI     0F0H
        JNZ     CLIP
DONE:   RET
;
;
;;;;;;;;;;;;;;;;;;;; DIVIDE ACC BY TWO ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
DIV2:   ADI     0
        JM      NEG
        CPI     1
        JZ      LAST
        RRC
        ACI     0           ; ADD CARRY TO ROUND UPWARD.
        ANI     7FH
        RET
NEG:    CPI     -1
        JZ      LAST
        RRC
        ORI     80H
        RET
LAST:   MVI     A,0
        RET
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;; LINE DRAWING ROUTINE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
DRAWL:  CALL    TRANL       ; APPLY TRANSFORMATION TO LINE.
        CALL    DELXY
        LDA     X1
        STA     XOUT
        LDA     Y1
        STA     YOUT
        CALL    PTOUT
        LDA     DELTX
        LRDISP  D,DELTY
        ORA     D
        RZ                  ;RETURN IF BOTH DELTAS=0 (PT PLOT).
        LDA     DELTX       ; FIND ABS(DELTX )
        CALL    MAG
```

```
        MOV     D,A
        LDA     DELTY       ; FIND ABS(DELTY )
        CALL    MAG
        MOV     E,A
CHEK:   ORA     D           ; CHECK FOR HOB SET (NORMALIZE).
        ANI     80H
        JNZ     NORM
        MOV     A,D         ; SHIFT LEFT DELTX  AND Y.
        SHLF    DUMMY
        MOV     D,A
        MOV     A,E
        SHLF    DUMMY
        MOV     E,A
        JMP     CHEK        ; CONTINUE SHIFTING UNTIL NORMALIZED.
NORM:   SRDISP  D,DELTX     ; THE DELTAS ARE NOW NORMALIZED.
        SRDISP  E,DELTY     ; CUR X OR Y IN H-L, CUR DELTA IN D-E.
        LRDISP  H,X1        ; INITIALIZE X AND Y.
        MVI      L,80H
        MVI     E,80H
        PUSH    H
        LRDISP  D,Y1
        POP     H
        PUSH    D           ; PUT INITIAL Y ON TOP OF STACK.
NXTPT:  PUSH    H
        LRDISP  E,DELTX
        MVI     D,0FFH
        LDA     SIGNX
        ADI     0           ;SET COND FLAGS
        JP      POSX
        MOV     A,E         ; COMPLEMENT ACC IF SIGN OF DELTAX
        CMA                 ;              IS NEGATIVE.
        INR     A
        MOV     E,A
        JMP     ADDX
POSX:   INR     D           ; CLEAR ALL BITS OF D.(SIGN OF DELTAX
ADDX:   POP     H           ;                  IS POSITIVE).
        DAD     D
        MOV     A,H
        STA     XOUT
        XTHL                ;EXCHANGE X FOR Y.
        PUSH    H
        LRDISP  E,DELTY
        MVI     D,0FFH
        LDA     SIGNY
        ADI     0           ; SET CONDITION FLAGS.
        JP      POSY
        MOV     A,E         ; COMPLEMENT ACC IF DELTAY IS NEGITIVE.
        CMA
        INR     A
        MOV     E,A
        JMP     ADDY
POSY:   INR     D           ; CLEAR ALL BITS OF D (SIGN OF DELTA
ADDY:   POP     H           ;              IS POSITIVE).
```

```
         DAD       D
         MOV       A,H
         STA       YOUT
         CALL      PTOUT       ; OUTPUT POINT.
         LDA       Y2          ; CHECK FOR REACHING POINT 2
         CMP       H
         JZ        XCHK
         XTHL
         JMP       NXTPT
XCHK:    XTHL
         PUSH      H
         LDA       X2
         POP       H
         CMP       H
         JNZ       NXTPT
         POP       H           ; RETURN WHEN POINT 2 IS REACHED
         RET
;
;
;;;
```

```
          MOV       D,A
          JMP       DTRAN
TMUL:     CALL      MULT
DTRAN:    MOV       M,C
          RET
 ;
 ;
 ;;;;;;;;;;;;;;;EXCHANGE POINTS 1 AND 2 ;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;
EXCH12:  LRDISP     D,X1
          LRDISP    E,X2
          SRDISP    D,X2
          SRDISP    E,X1
          LRDISP    D,Y1
          LRDISP    E,Y2
          SRDISP    E,Y1
          SRDISP    D,Y2
          RET
 ;
 ;
 ;;;;;;;;;;;;;;;;;;; CALCULATE KEYS FOR POINTS 1 AND 2 ;;;;;;;;;;;;;
 ;
CKEY12:  LRDISP     D,X1
          LRDISP    E,Y1
          CALL      CKEY
          STA       KEY1
          LRDISP    D,X2
          LRDISP    E,Y2
          CALL      CKEY
          STA       KEY2
          RET
 ;
 ;
 ;;;;;;;;;;;; CALCULATE KEY FOR TEMPORARY POINT;;;;;;;;;;;;;;;;;;
 ;
CKEYT:   LRDISP     D,X1
          LRDISP    E,Y1
          CALL      CKEY
          STA       KEYT
          RET
 ;
 ;
 ;;;;;;;;;;;;;;;;;;;;;;;; CALCULATE KEY ;;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;
CKEY:    MVI       H,0          ; POINT IS IN D-E KEY IN ACC.
          MVI       L,0          ; KEY WILL BE FORMED IN L, THE B-KEY
                                 ; WILL BE FORMED IN H.
          PUSH      H
          LDA       TYLIM        ; TEST TOP LIMIT.
          POP       H
          CMP       E
          CALL      INCHGH       ; INCREMENT REG IF GREATER THAN LIMIT.
          DAD       H            ;SHIFT H-L LEFT.
```

```
        PUSH    H
        LDA     BYLIM       ;TEST FOR BOTTOM LIMIT.
        POP     H
        CMP     E
        CALL    INCLOW      ; INCREMENT REG IF LESS THAN LIMIT.
        DAD     H
        PUSH    H
        LDA     RXLIM       ; TEST FOR RIGHT LIMIT.
        POP     H
        CMP     D
        CALL    INCHGH
        DAD     H
        PUSH    H
        LDA     LXLIM       ; TEST FOR LEFT LIMIT.
        POP     H
        CMP     D
        CALL    INCLOW
        MOV     A,H
        RFOUP   DUMMY
        OSA     L
        RET
;
;
;;;;;;;;;; DETERMINE THE SIGN OF THE NEXT X INCREMENT ;;;;;;;;;;
;
CSIGN:  LRDISP  D,DELTK
        LDA     KEY1
        LRDISP  H,KEYT
        ANA     H
        ANI     0FH
        JNZ     AWAY
TOWRD:  LDA     SIGNK       ; SET SIGN SO THAT THE TEMP POINT
        XRA     D           ; MOVES CLOSER TO POINT 1.
        ANI     80H
        JZ      COMPD       ; JUMP IF SIGN BITS =.
        RET
AWAY:   LDA     SIGNK       ; SET SIGN SO THAT THE TEMP POINT
        XRA     D           ; MOVES AWAY FROM POINT 1.
        ANI     80H
        JNZ     COMPD       ; JUMP IF SIGN BITS NOT =.
        RET
COMPD:  LDA     DELTY       ; COMPLEMENT ALL THREE DELTAS.
        CMA
        INR     A
        STA     DELTX
        LDA     DELTY
        CMA
        INR     A
        STA     DELTY
        LDA     DELTK
        CMA
        INR     A
        STA     DELTK
```

```
        RET
;
;
;;;;;;;;;; CALCULATE DELTA X AND DELTA Y ;;;;;;;;;;;;;;;;;;;;
;
DELXY:  LDA     X2          ; CALCULATE DELTX .
        LRDISP  D,X1
        SUB     D
        STA     DELTX
        MVI     E,1
        JP      SPX         ; DETERMINE THE SIGN OF DELTA X.
        MVI     E,-1
SPX:    SRDISP  E,SIGNX
        LDA     Y2          ; CALCULATE DELTA Y.
        LRDISP  D,Y1
        SUB     D
        STA     DELTY
        MVI     E,1
        JP      SPY         ; DETERMINE THE SIGN OF DELTA Y.
        MVI     E,-1
SPY:    SRDISP  E,SIGNY
        RET
;
;
;;;;;;;;;;;;;;;; CALC KEY SIGN AND DELT ;;;;;;;;;;;;;;;;;;;;;;
;
KYSGN:  LDA     DELTX       ; THE KEY SIGN AND DELTA ARE USED TO
        CALL    MAG         ; INSURE THAT THE DIRECTIONS OF THE
        MOV     D,A         ; ORIGINAL DELTA AND THE MOST RECENT
        LDA     DELTY       ; DELTA ARE SAVED.
        CALL    MAG
        MVI     E,1
        CMP     D
        JC      XHIGH
        LDA     DELTY
        ADI     0
        JP      SETK
        MVI     E,-1
        JMP     SETK
XHIGH:  LDA     DELTX
        ADI     0
        JP      SETK
        MVI     E,-1
SETK:   SRDISP  E,SIGNK
        STA     DELTK
        RET
;
;
;;;;;;;; INCREMENT POINT 1 BY DELTA X AND DELTA Y ;;;;;;;;;;;;;;
;
ADDEL:  LDA     X1
        LRDISP  D,DELTX
        ADD     D
```

```
        STA     X1
        LDA     Y1
        LRDISP  D,DELTY
        ADD     D
        STA     Y1
        RET
;
;
;;;;;;;;;;;;;;;; TAKE ABSOLUTE VALUE OF ACC ;;;;;;;;;;;;;;;;;;;;
;
MAG:    ADI     0           ;SET COND FLAGS.
        RP                  ; RETURN IMMED IF + OTHERWISE COMPL.
        CMA
        INR     A
        RET
;
;
;;;;;;;;;;;;;;;; INCREMENT IF ABOVE LIMIT ;;;;;;;;;;;;;;;;;;;;;;
;
INCHGH: JNZ     N           ; INCREMENT H IF ACC=REG.
        INR     H           ; INCREMENT L IF ACC>=REG.
        INR     L
        RET
N:      RNC
        INR     L
        RET
;
;
;;;;;;;;;;;;;;;; INCREMENT IF BELOW LIMIT ;;;;;;;;;;;;;;;;;;;;;;
;
INCLOW: JNZ     MMM         ; INCREMENT H IF ACC=REG.
        INR     H           ; INCREMENT L IF ACC<=REG.
        INR     L
        RET
MMM:    RC
        INR     L
        RET
;
;
;;;;;;;;;;;;;;;;;;;; OUTPUT A POINT ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PTOUT:  PUSH    H           ; SAVE ALL REGISTERS.
        PUSH    L
        PUSH    B
        LHLD    POUT        ; GET ADDRESS OF POINT TO BE OUTPUT.
        MVI     A,80H
        ORA     H
        MOV     H,A         ; CONVERT REL TO ACTUAL ADDRESS.
        MOV     A,L         ; DETERMINE THE MASK TO USE.
        ANI     07H
        MVI     B,00H
        MOV     C,A
        LXI     D,BTMSK     ; GET ADDRESS OF MASK TABLE.
```

```
        XCHG
        DAD     B           ; COMPUTE ADDRESS IN MASK TABLE.
        MOV     C,M         ; C <- BIT MASK.
        XCHG
        MOV     B,M         ; B <- DATA BYTE.
        LDA     STATUS
        ANI     FLMOD       ; TEST FOR NORM OF REV FIELD.
        JNZ     REVPT
        MOV     A,C
        ORA     B
        JMP     PTO1
REVPT:  MOV     A,C
        CMA                 ; REVERSE THE MASK.
        ANA     B
PTO1:   MOV     M,A         ; REPLACE BYTE.
        POP     B           ; RESTORE REGISTERS.
        POP     D
        POP     H
        RET
BTMSK:  DB      01H
        DB      02H
        DB      04H
        DB      08H
        DB      10H
        DB      20H
        DB      40H
        DB      80H
;
;
;
;;;
```

```
        JMP     DCHR1
NLF:    CPI     20H             ; CHECK FOR SPACE.
        JZ      DCHR2
NSF:    CALL    CHMAP           ; MAP ASCII VALUE TO CGEN ADDRESS.
        PUSH    PSW             ; SAVE CHARACTER CODE.
        MOV     A,L
        CPI     LOIN            ; CHECK FOR NO MORE ROOM ON LINE.
        JC      CHAR1           ; JUMP IF ACC<78.
        MVI     A,00H           ; SET YPOS TO 0 (CARRIGE RETURN).
        MOV     L,A
        CALL    PROLF           ; PROCESS LINE FEED.
CHAR1:  MVI     A,00H
        STA     LPCN1           ; ZERO LOOP COUNTER.
        CALL    CHMSK           ; BC <- CHARACTER MASK.
        POP     PSW             ; GET CHARACTER TO BE DISPLAYED.
        MOV     E,A
        ANA     A               ; SET FLAGS.
        JP      STCH            ; IF HOB IS 0 THEN STAND. CHAR.
        ANI     7FH             ; HOB WAS 1 SO CHAR IS USER DEFINED.
        RLC
        RLC
        RLC                     ; MULTIPLY CHAR CODE BY 8 TO
        ANI     0F8H            ; OBTAIN OFFSET INTO RAM.
        LXI     D,UCHB          ; GET BASE OF USER CHAR AREA.
        ADD     E               ; ADD OFFSET INTO USER CHAR AREA.
        MOV     E,A             ; DE <- CHARACTER ADDRESS (USER DEF.)
        JMP     LROW
STCH:   MVI     D,CHARH         ; DE <- CHARACTER ADDRESS (STANDARD)
        CPI     40H
        JNC     LROW            ; JUMP IF CHAR. IS UPPER CASE.
        MVI     A,02
        STA     LPCN1           ; DECREASE ROWS IN CHAR BY 2.
        INR     H
        INR     H
LROW:   LDAX    D               ; GET ROW OF CHARACTER.
        ANI     7FH             ; CLEAR HOB'S.
        PUSH    D               ; SAVE CHARACTER ADDRESS.
        CALL    OUTRO           ; OUTPUT A ROW OF THE CHARACTER.
        POP     D               ; RESTORE CHARACTER ADDRESS.
        MOV     A,D             ; GET HOB OF CHAR ADDRESS.
        CPI     08H             ; DETERMINE IF CHAR IS STAND.
        JNZ     STCH2           ; JUMP IF CHAR IS STANDARD.
        INR     E               ; SET CHAR ADDRESS TO NEXT ROW OF CHAR.
        JMP     LROW2
STCH2:  INR     D               ; SET CHAR ADDRESS TO NEXT ROW OF CHAR.
LROW2:  INR     H               ; INCREMENT THE Y SCREEN POSITION.
        LDA     LPCN1
        INR     A
        STA     LPCN1           ; INCREMENT LOOP COUNTER.
        CPI     08H             ; CHECK FOR END OF LOOP.
        JNZ     LROW
;
DCHR2:  LHLD    CHPOS           ; OBTAIN CHARACTER POSITION.
```

```
          MOV       A,L
          ADI       06H         ; UPDATE CURSOR POSITION.
          ANI       7FH
          MOV       L,A
DCHR1:    SHLD      CHPOS       ; UPDATE RAM CHAR POSITION.
          CALL      PUTCUR      ; UPDATE CRTC CURSOR.
          RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; OUTPUT ONE ROW OF A CHARACTER. ON ENTRY THE REGISTERS SHOULD  ;
; BE AS FOLLOWS-                                               ;
;       BC <- CHARACTER MASK.                                  ;
;      ACC <- THE ROW OF THE CHARACTER.                        ;
;       HL <- SCREEN ADDRESS WHERE ROW IS TO BE PLACED.        ;
;       DE    IS USED BY THIS ROUTINE SO ITS CONTENTS SHOULD   ;
;             BE SAVED BY THE CALLING PROGRAM.                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
OUTPO:    MVI       D,0
          MOV       E,A         ; DE <- THE ROW OF THE CHARACTER.
          LDA       STATUS
          ANI       CHMOD       ; DETERMINE IF NORMAL OR REV. CHAR MODE.
          JZ        DROW
          MOV       A,E         ; IF REVERSE MODE, COMPLEMENT CHAR ROW.
          CMA
          ANI       1FH
          MOV       E,A
DRCW:     CALL      SHCHR       ; SHIFT CHARACTER INTO POSITION.
          MOV       A,M         ; GET SCREEN BYTE.
          ANA       C           ; CUT WINDOW FOR CHARACTER.
          ORA       E           ; OR IN CHARACTER ROW.
          MOV       M,A         ; REPLACE SCREEN BYTE.
          MOV       A,B
          CMA                   ; TEST FOR X-ING BYTE BNDRY. ON SCREEN.
          ANA       A           ; SET FLAGS.
          JZ        DRCW2       ; JUMP IF NO BOUNDRY CROSSED.
          MOV       A,L
          ADI       08H
          MOV       L,A
          MOV       A,M         ; FETCH NEXT BYTE FROM SCREEN.
          ANA       B           ; CUT WINDOW FOR CHARACTER.
          ORA       D           ; OR IN REMAINDER OF CHARACTER.
          MOV       M,A         ; REPLACE SCREEN BYTE.
          MOV       A,L         ; GET SCREEN ADDRESS.
          SUI       08H
          MOV       L,A
DRCW2:    RET
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PROCESS A LINE FEED. ON ENTRY HL CONTAINS THE CURSOR                    ;
; ADDRESS. THE QUANTITY IS UPDATED AS NECESSARY BY THIS                   ;
; ROUTINE ALONG WITH THE CHAR POSITION AND THE CURSOR.                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PRCLF:  MOV     A,H
        ADI     08H             ; INCREASE Y POSITION.
        PUSH    PSW             ; SAVE CY FLAG.
        ORI     80H
        MOV     H,A
        SHLD    CHPOS           ; UPDATE CHAR POSITION.
        CALL    PUTCUR          ; UPDATE CURSOR.
        POP     PSW             ; GET CY TO DET. IF PGFUL SET.
        JNC     NOPG
        LDA     STATUS          ; READ STATUS.
        ORI     PGFUL           ; SET PAGE FULL FLAG.
        STA     STATUS
NOPG:   LDA     STATUS
        ANI     PGFUL           ; CHECK PAGE FULL FLAG.
        JZ      NOPG2           ; JUMP IF END OF PAGE NOT YET REACHED.
        MVI     A,STABH
        STA     CR1A
        LDA     SALDH           ; GET HOB OF START ADDRESS.
        ADI     02H
        ANI     1FH             ; INSURE HO 3 BITS ARE CLEARED.
        STA     SALDH           ; UPDATE HOB OF START ADDRESS.
        STA     CR1D
NOPG2:  RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; MAP THE ASCII CHARACTER VALUE INTO THE CHARACTER ADDRESS                 ;
; SPACE. ONLY THE ACCUMULATOR IS ALTERED.                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
CHMAP:  ANA     A               ; SET FLAGS.
        RM                      ; RETURN IF USER DEFINED CHARACTER.
        CPI     60H
        JC      UPP
        SUI     60H
        RET                     ; RETURN FOR LOWER CASE.
UPP:    CPI     40H
        RNC                     ; RETURN FOR UPPER CASE ALPHA.
        ADI     40H
        RET                     ; RETURN FOR UPPER CASE #'S.
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PLACE THE VALUE GIVEN BY HL INTO THE CRTC CURSOR REG.                    ;
; IN ORDER TO DO THIS THE VALUE MUST BE PROPERLY PACKED.                   ;
; NO REGISTORS ARE CHANGED BY THIS ROUTINE OTHER THAN THE                  ;
; ACC.                                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PUTCUR:  PUSH    H
         MOV     A,L
         ADI     06H
         MOV     L,A
         MOV     A,H
         ADI     07H
         MOV     H,A
         RAR                      ; LOB NOW IN CY.
         MOV     H,A
         MOV     A,L
         JNC     NOCY
         ORI     80H              ; SET HOB.
         JMP     PUT1
NOCY:    ANI     7FH              ; CLEAR HOB.
PUT1:    MOV     L,A
         MOV     A,H              ; SHIFT HL RIGHT ONE.
         RAR
         MOV     H,A
         MOV     A,L
         RAR
         MOV     L,A
         MOV     A,H
         ANI     1FH              ; CLEAR 3 HOB'S.
         MOV     H,A
         MVI     A,CURSH
         STA     CRTA
         MOV     A,H
         STA     CRTD             ; SET CURSOR (H).
         MVI     A,CURSL
         STA     CRTA
         MOV     A,L
         STA     CRTD             ; SET CURSOR (L).
         POP     H
         RET
```

```
;;;
```

```
;;;
```

```
          CALL      PROBYT
;
ERCW2:    MOV       L,C         ; RESET X POS TO BEGIN NEW ROW.
          INR       H           ; INCREASE Y POSITION.
          MOV       A,H
          CMP       D           ; CHECK FOR LAST ROW.
          JM        REP1
          JZ        REP1
          RET
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  PROCESS A BYTE ACCORDING TO THE VALUE OF THE FLAG IN                      ;
;  LOCATION CLCMP.   IF THE FLAG IS FF THEN (HL) IS TO                       ;
;  BE CLEARED (TO ZERO OR 1 DEPENDING ON THE FIELD MODE.).                   ;
;  IF THE FLAG IS ZERO, (HL) IS TO BE COMPLEMENTED. THE                      ;
;  PART OF THE BYTE TO BE PROCESSED IS GIVEN BY THE POS-                     ;
;  ITION OF THE 1'S IN B. NO REGISTERS OTHER THAN ACC ARE                    ;
;  ALTERED BY THIS ROUTINE.                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PROBYT:   PUSH      H           ; SAVE SCREEN POSITION.
          MOV       A,H
          ORI       80H
          MOV       H,A         ; CONVERT REL TO ACTUAL ADDRESS.
          LDA       CLCMP       ; READ CLEAR/COMPLEMENT FLAG.
          ANA       A           ; SET FLAGS.
          JNZ       CLR1        ; CLEAR BYTE IF FLAG IS NON-ZERO.
          MOV       A,B         ; COMPLEMENT THE BYTE.
          XRA       M           ; XOR MASK WITH MEMORY.
          JMP       DPROB
CLR1:     LDA       STATUS
          ANI       FLMOD       ; CHECK FIELD MODE.
          JZ        FLD0
          MOV       A,M
          ORA       B           ; SET DESIRED BITS TO 1.
          JMP       DPROB
FLD0:     MOV       A,B
          CMA                   ; COMPLEMENT MASK.
          ANA       M           ; CLEAR DESIRED BITS.
DPROB:    MOV       M,A         ; REPLACE BYTE.
          POP       H
          RET
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; MULTIPLY ROUTINE. MULTIPLIES THE CONTENTS OF C BY THE ;
; CONTENTS OF D. THE RESULT IS RETURNED IN BC. E IS USED;
; AS A LOOP COUNTER, HL IS UNCHANGED. PROGRAM  USED IS ;
; TAKEN FROM INTEL 8080 ASSEMBLY LANGUAGE MANUAL.      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MULT:    MVI     B,0
         MVI     E,9
MULT0:   MOV     A,C
         RAR
         MOV     C,A
         DCR     E
         JZ      DMUL
         MOV     A,B
         JNC     MULT1
         ADD     D
MULT1:   RAR
         MOV     B,A
         JMP     MULT0
DMUL:    RET
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; DIVIDE ROUTINE. DIVIDES THE CONTENTS OF C BY THE CON- ;
; TENTS OF D. THE RESULT COMES BACK IN C. E IS USED FOR ;
; A LOOP COUNTER. HL ARE NOT CHANGED.                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DIV:     MVI     B,0
         MOV     A,E
         MVI     E,9
DIV0:    SUB     D
         JC      DIV1
         MOV     B,A
DIV1:    CMC
         MOV     A,C
         RAL
         MOV     C,A
         MOV     A,E
         RAL
         MOV     B,A
         DCR     E
         JNZ     DIV0
         RET
```

```
;;;;;;;;;;;; CPTC ADJUSTMENT ROUTINE ;;;;;;;;;;;;
CRTAJ:   CALL     GETDAT    ; SPECIFY THE REG TO BE CHANGED.
         STA      CRTA
         CALL     GETDAT    ; SPECIFY THE NEW VALUE.
         STA      CRTD
         RET
;
;;;;;;;;;;;;;;;; CRT INITIALIZATION TABLE ;;;;;;;;;;;;;;;;;;
CRTTAB:  DB       52H       ; HORIZ TOTAL.
         DB       40H       ; HORIZ DISPLAYED.
         DB       45H       ; HORIZ SYNC POSITION.
         DB       06H       ; HORIZ SYNC WIDTH.
         DB       7FH       ; VERT TOTAL.
         DB       10H       ; VERT TOTAL ADJUST.
         DB       7FH       ; VERT DISPLAYED.
         DB       7FH       ; VERT SYNC POSITION.
         DB       00H       ; INTERLACE MODE.
         DB       01H       ; MAX SCAN LINE IN CHARACTER.
         DB       40H       ; CURSOR ENABLE AND ST WITHIN CHAR.
         DB       01H       ; CURSOR END WITHIN CHAR.
         DB       00H       ; MEM START ADDRESS (H).
         DB       00H       ; MEM START ADDRESS (L).
         DB       00H       ; CURSOR ADDRESS (H).
         DB       00H       ; CURSOR ADDRESS (L).
END
```

A.2 Main Processor Driver Listing

```
STATUS  EQU     70H
DATPT   EQU     STATUS+1
COMPT   EQU     STATUS+2
NCOM    EQU     22
TABLN   EQU     NCOM*2
COMLN   EQU     06
ISTR    EQU     4F90H
LPCN1   EQU     4F8FH
LPCN2   EQU     4F8EH
STACK   EQU     4F80H
EXADD   EQU     3800H
;
;
        ORG     5000H       ; BEGIN ASSEMBLY AT 5000H
        LXI     SP,STACK    ; INITIALIZE STACK POINTER.
MLOOP:  CALL    CRLF
        LXI     H,COMD
        CALL    PTSTR       ; REQUEST COMMAND.
        CALL    INSTR       ; FETCH INPUT STRING.
        MVI     A,00
        STA     LPCN2
        LXI     H,STLOC     ; HL <- LOCATION OF COMMAND STRINGS.
        LXI     B,COMLN+1   ; BC <- LENGTH OF A COMMAND STRING.
SEARCH: CALL    CMPST       ; COMPARE IN STR. WITH NEXT COM. STR.
        ANA     A
        JZ      MATCH
        DAD     B
        LDA     LPCN2
        ADI     02H
        STA     LPCN2
        CPI     TABLN
        JNZ     SEARCH
        LXI     H,UNDEF     ; INPUT STRING NOT A COMMAND STRING.
        CALL    PTSTR
        CALL    CRLF
        JMP     MLOOP
MATCH:  LDA     LPCN2
        MOV     E,A
        MVI     D,0         ; DE <- OFFSET INTO TABLES.
        LXI     H,TITL      ; OBTAIN ADDR OF COMMAND TITLE.
        DAD     D
        MOV     C,M
        INX     H
        MOV     H,M
        MOV     L,C
        CALL    PTSTR       ; PRINT COMMAND TITLE.
        CALL    CRLF
        LXI     H,JMPTAB    ; OBTAIN COMMAND ROUTINE ADDR.
```

```
          DAD       D
          MOV       E,M
          INX       H
          MOV       D,M
          XCHG
          LXI       B,MLOOP
          PUSH      B           ; PLACE RETURN ADDR ON STACK.
          PCHL                  ; JUMP THROUGH HL.
;
;;;;;;;;;;;;;;;; JUMP TABLE ;;;;;;;;;;;;;;;;;
JMPTAB:   DW        INIT
          DW        DVEC
          DW        WINX
          DW        WINY
          DW        DRAW
          DW        DRAC
          DW        SCAX
          DW        SCAY
          DW        SHFX
          DW        SHFY
          DW        CHRS
          DW        SCRV
          DW        HOME
          DW        CURP
          DW        CLER
          DW        CLEZ
          DW        COMP
          DW        COMZ
          DW        EXIT
          DW        HELP
          DW        DEFCH
          DW        CRTAJ
;
;;;;;;;;;;;;;;;; COMMAND ROUTINES ;;;;;;;;;;;;;;;;
INIT:     MVI       A,00
          OUT       COMPT
          MVI       A,01
          OUT       STATUS
 IWAIT:   IN        STATUS      ; WAIT FOR INITAILIZATION TO FINISH.
          ANI       07H
          JNZ       IWAIT
          RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DVEC:     MVI       A,07H
          CALL      SENDCOM
          CALL      XYCOR
          CALL      XYCOR
          RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WINX:     MVI       A,0BH
          CALL      SENDCOM
          CALL      RANGE
          RET
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WINY:   MVI     A,09H
        CALL    SENDCOM
        CALL    RANGE
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DRAW:   MVI     A,0AH
        CALL    SENDCOM
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DRAC:   MVI     A,0BH
        CALL    SENDCOM
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCAX:   MVI     A,0CH
        CALL    SENDCOM
        CALL    FACTOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCAY:   MVI     A,0DH
        CALL    SENDCOM
        CALL    FACTOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SHEX:   MVI     A,0EH
        CALL    SENDCOM
        CALL    FACTOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SHFY:   MVI     A,0FH
        CALL    SENDCOM
        CALL    FACTOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CHRS:   MVI     A,11H
        CALL    SENDCOM
 CLP:   CALL    CHIN
        CPI     00H
        JZ      DCH
        CPI     1BH             ; CHECK FOR USER CHAR.
        JZ      USDF
        MOV     B,A
        CALL    CHOUT
        MOV     A,B
        CALL    SENDAT
        JMP     CLP
 USDF:  CALL    GETDIG
        ORI     80H
        CALL    SENDAT
        JMP     CLP
 DCH:   CALL    SENDAT
        CALL    CRLF
        RET
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCRV:   MVI     A,14H
        CALL    SENDCOM
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
HOME:   MVI     A,15H
        CALL    SENDCOM
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CURP:   MVI     A,16H
        CALL    SENDCOM
        CALL    XYCOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CLER:   MVI     A,17H
        CALL    SENDCOM
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CLEZ:   MVI     A,18H
        CALL    SENDCOM
        CALL    XYCOR
        CALL    XYCOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CCMP:   MVI     A,19H
        CALL    SENDCOM
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
COMZ:   MVI     A,1AH
        CALL    SENDCOM
        CALL    XYCOR
        CALL    XYCOR
        RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EXIT:   CALL    CRLF
        CALL    CRLF
        CALL    CRLF
        JMP     EXADD
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DEPCH:  MVI     A,13H
        CALL    SENDCOM
        LXI     H,CHNS
        CALL    PTSTR
        CALL    GETDIG
        CALL    SENDAT
        CALL    CRLF
        LXI     H,CHBS
        CALL    PTSTR
        CALL    CRLF
        MVI     A,08H
        STA     LPCN1
 ULP1:  MVI     A,05H
        STA     LPCN2
```

```
         MVI     D,0         ; GET THE CHARACTER.
ULP2:    CALL    GETDIG      ; GET A ROW OF THE CHARACTER.
         STC
         JNZ     ROT1        ; ROTATE 1 INTO D.
         CMC                 ; ROTATE 0 INTO D.
ROT1:    MOV     A,D
         RAR
         MOV     D,A
         LDA     LPCN2
         DCR     A
         STA     LPCN2
         JNZ     ULP2
         MOV     A,D
         RRC
         RRC
         RRC
         CALL    SENDAT
         CALL    CRLF
         LDA     LPCN1
         DCR     A
         STA     LPCN1
         JNZ     ULP1
         RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
HELP:    MVI     A,00
         STA     LPCN1       ; INITIALIZE LOOP COUNTER
         LXI     H,STLOC     ; HL <- COMMAND NAME STRING LOC.
         LXI     D,TITL      ; DE <- COMMAND DESCRIPTION LOC.
HLOP:    CALL    PTSTR
         PUSH    H
         LXI     H,BLANK
         CALL    PTSTR       ; PRINT BLANKS
         LDAX    D
         MOV     L,A
         INX     D
         LDAX    D
         MOV     H,A
         INX     D
         CALL    PTSTR
         CALL    CRLF
         POP     H
         LDA     LPCN1
         ADI     02
         STA     LPCN1
         CPI     TABLN
         JZ      DHELP
         MVI     B,0
         MVI     C,CCMLN+1
         DAD     B
         JMP     HLOP
DHELP:   RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CRTAJ:   MVI     A,1BH
```

```
          CALL      SENDCOM
          CALL      GETHEX      ; SPECIFY REGISTER TO BE CHANGED.
          CALL      SENDAT
          CALL      CRLF
          CALL      GETHEX      ; SPECIFY NEW VALUE.
          CALL      SENDAT
          CALL      CRLF
          RET
;
;;;;;;;;;; REQUEST XY COOR. PAIR ;;;;;;;;;;
XYCOR:    LXI       H,XMES
          CALL      PTSTR
          CALL      GETHEX
          JP        XOK         ; CHECK FOR LEGAL VALUE.
          CALL      ERROR
          JMP       XYCOR
 XCK:     CALL      SENDAT
 YCOR:    LXI       H,YMES
          CALL      PTSTR
          CALL      GETHEX
          JP        YOK         ; CHECK FOR LEGAL VALUE.
          CALL      ERROR
          LXI       H,BL17
          CALL      PTSTR
          JMP       YCOR
 YCK:     CALL      SENDAT
          CALL      CRLF
          RET
;
;;;;;;;;;; REQUEST RANGE ;;;;;;;;
RANGE:    LXI       H,BRANG
          CALL      PTSTR
          CALL      GETHEX
          JP        XRCK        ; CHECK FOR LEGAL VALUE.
          CALL      ERROR
          JMP       RANGE
 XROK:    CALL      SENDAT
 YRAN:    LXI       H,ERANG
          CALL      PTSTR
          CALL      GETHEX
          JP        YROK        ; CHECK FOR LEGAL VALUE.
          CALL      ERROR
          LXI       H,BL23
          CALL      PTSTR
          JMP       YRAN
 YPOK:    CALL      SENDAT
          CALL      CRLF
          RET
;
;;;;;;;;;; REQUEST FACTOR ;;;;;;;;;;
FACTOR:   LXI       H,FACT
          CALL      PTSTR
          CALL      GETHEX
```

```
          CALL      SENDAT
          CALL      CRLF
          RET
;
;;;;;;;;;; PRINT A STRING BEGINNING AT ADDR. HL ;;;;;;;;;;
;;;;;;;;;; HL CHANGED.
PTSTR:    PUSH      B
          PUSH      H
 PTLP:    MOV       B,M
          MOV       A,B
          CPI       '$'
          JZ        DNPTL
          CALL      CHOUT
          INX       H
          JMP       PTLP
 DNPTL:   POP       H
          POP       B
          RET
;
;;;;;;;;;; INPUT A STRING OF CHARS ;;;;;;;;;;
;;;;;;;;;; STRING IS PLACED AT LOC ISTR.
INSTR:    PUSH      H
          PUSH      B
          LXI       H,ISTR
          MVI       A,COMLN
          STA       LPCN1
 CLRS:    MVI       A,20H      ; CLEAR STRING AREA.
          MOV       M,A
          INX       H
          LDA       LPCN1
          DCR       A
          STA       LPCN1
          JNZ       CLRS
          LXI       H,ISTR
 ISLP:    CALL      CHIN
          CPI       0DH
          JZ        DINS
          MOV       B,A
          CALL      CHOUT
          MOV       A,B
          MOV       M,A
          INX       H
          JMP       ISLP
 DINS:    CALL      CRLF
          POP       B
          POP       H
          RET
;;;;;;;;;; COMPARE TWO CHARACTER STRINGS ;;;;;;;;;;
;;;;;;;;;; CONSTANT STRING ADDR IN DE
CMPST:    PUSH      B
          PUSH      D
          PUSH      H
          LXI       D,ISTR
```

```
          MVI       C,0
          MVI       A,COMLN
          STA       LPCN1
 CCLP:    MOV       B,M
          LDAX      D
          CMP       B
          JNZ       RFAIL
          LDA       LPCN1
          INX       D
          INX       H
          DCR       A
          STA       LPCN1
          JNZ       COLP
          JMP       RSUC
 RFAIL:   MVI       C,-1
 RSUC:    MOV       A,C
          POP       H
          POP       D
          POP       B
          RET
;
;;;;;;;;;;;; EPROR- VALUE OUT OF RANGE ;;;;;;;;;;;;
ERROR:    LXI       H,ERRS
          CALL      PTSTR
          CALL      CRLF
          RET
;
GETHEX:   PUSH      B
          CALL      GETDIG
          RLC
          RLC
          RLC
          RLC
          MOV       C,A
          CALL      GETDIG
          ORA       C
          POP       B
          RET
;
GETDIG:   CALL      CHIN
          MOV       B,A
          CALL      CHOUT
          MOV       A,B
          CPI       40H
          JNC       HCHR
          ANI       0FH
          RET
HCHR:     ADI       09H
          ANI       0FH
          RET
;
OUTHEX:   PUSH      D
          MOV       D,A
```

```
          ANI       0F0H
          RRC
          RRC
          RRC
          RRC
          CALL      OUTDIG
          MOV       A,D
          ANI       0FH
          CALL      OUTDIG
          MOV       A,D
          POP       D
          RET
;
OUTDIG:   CPI       0AH
          JC        DOUT
          ADI       37H
          JMP       DOUT2
DOUT:     ADI       30H
DOUT2:    MOV       B,A
          CALL      CHOUT
          RET
;
;;;;;;;;;; SEND A COMMAND TO THE MPIX ;;;;;;;;;;
SENDCOM:  OUT       COMPT
          IN        STATUS
          ORI       01H
          OUT       STATUS
CWAIT:    IN        STATUS
          ANI       02H
          JZ        CWAIT
          IN        STATUS
          ANI       0FDH
          OUT       STATUS
          RET
;
;;;;;;;;;; SEND DATA TO THE MPIX ;;;;;;;;;;
SENDAT:   OUT       DATPT
          IN        STATUS
          ORI       01H
          OUT       STATUS
DWAIT:    IN        STATUS
          ANI       02H
          JZ        DWAIT
          IN        STATUS
          ANI       0FDH
          OUT       STATUS
          RET
;
CRLF:     MVI       B,0DH
          CALL      CHOUT
          MVI       B,0AH
          CALL      CHOUT
          RET
```

```
;
CHCUT:  IN      3
        ANI     01
        JZ      CHOUT
        MOV     A,B
        OUT     2
        RET
;
CHIN:   IN      3
        ANI     02H
        JZ      CHIN
        IN      2
        ANI     7FH
        RET
;
DISPA:  PUSH    D
        PUSH    B
        MOV     D,A
        CALL    OUTHEX
        CALL    CHIN
        CALL    CRLF
        MOV     A,L
        POP     B
        POP     D
        RET
;
;;;;;;;;;; COMMAND STRINGS ;;;;;;;;;;
STIOC:  DB      'INIT  '
        DB      'VECTOR'
        DB      'WINDX '
        DB      'WINDY '
        DB      'DRAW  '
        DB      'DRAWC '
        DB      'SCALEX'
        DB      'SCALEY'
        DB      'SHIFTX'
        DB      'SHIFTY'
        DB      'CHARS '
        DB      'SCRLV '
        DB      'HOME  '
        DB      'CURPOS'
        DB      'CLEAR '
        DB      'CLEARR'
        DB      'COMP  '
        DB      'COMPR '
        DB      'EXIT  '
        DB      'HELP  '
        DB      'DEFCH '
        DB      'CRTADJ'
```

```
;;;;;;;;;;; MESSAGE TABLE ;;;;;;;;;;
XMES:    DB        'X COORDINATE ->$'
YMES:    DB        '      Y COORDINATE ->$'
COMD:    DB        'ENTER COMMAND ->$'
UNDEF:   DB        'UNDEFINED COMMAND, TRY AGAIN.$'
BRANG:   DB        'LOWER END OF RANGE ->$'
ERANG:   DB        '      UPPER END OF RANGE ->$'
FACT:    DB        'FACTOR ->$'
ERRS:    DB        ' <- TOO LARGE, TRY AGAIN.$'
CHNS:    DB        'ENTER THE CHARACTER NUMBER ->$'
CHBS:    DB        'ENTER THE 8 ROWS OF THE CHARACTER BELOW...$'
BL17:    DB        '                  $'
BL23:    DB        '                        $'
BLANK:   DB        '        $'
;
;;;;;;;;;;;;;;;; TITLE TABLE ;;;;;;;;;;;;;;;;;
TITL:    DW          INITS
         DW          DVECS
         DW          WINXS
         DW          WINYS
         DW          DRAWS
         DW          DRACS
         DW          SCAXS
         DW          SCAYS
         DW          SHFXS
         DW          SHFYS
         DW          CHRSS
         DW          SCRVS
         DW          HOMES
         DW          CURPS
         DW          CLERS
         DW          CLEZS
         DW          COMPS
         DW          COMZS
         DW          EXITS
         DW          HELPS
         DW          DEFCS
         DW          CRTAS
;
;;;;;;;;;; TITLE STRINGS ;;;;;;;;;;
INITS:   DB        'INITIALIZE MPIX$'
DVECS:   DB        'DEFINE A VECTOR$'
WINXS:   DB        'DEFINE THE WINDOW IN THE X DIRECTION$'
WINYS:   DB        'DEFINE THE WINDOW IN THE Y DIRECTION$'
DRAWS:   DB        'DISPLAY THE CURRENTLY DEFINED VECTOR$'
DRACS:   DB        'CLIP AND DISPLAY THE CURRENTLY DEFINED VECTOR$'
SCAXS:   DB        'DEFINE THE X SCALING FACTOR$'
SCAYS:   DB        'DEFINE THE Y SCALING FACTOR$'
SHFXS:   DB        'DEFINE THE AMOUNT OF SHIFT IN THE X DIRECTION$'
SHFYS:   DB        'DEFINE THE AMOUNT OF SHIFT IN THE Y DIRECTION$'
CHRSS:   DB        'ENTER CHARACTER MODE$'
SCRVS:   DB        'REVERSE THE SCREEN AND CHARACTER MODES$'
HOMES:   DB        'RESET CHARACTER PAGE PARAMETERS$'
```

```
CUPPS:    DB        'DEFINE CURSOR POSITION$'
CLERS:    DB        'CLEAR THE SCREEN$'
CLEZS:    DB        'CLEAR A REGION OF THE SCREEN$'
CCMPS:    DB        'COMPLEMENT THE SCREEN$'
CCMZS:    DB        'COMPLEMENT A REGION OF THE SCREEN$'
EXITS:    DB        'EXIT THE DRIVER. CONTROL PASSED TO AMS$'
HELPS:    DB        'PRINT THE COMMAND LISTING$'
DEFCS:    DB        'DEFINE A USER CHARACTER$'
CRTAS:    DB        'CHANGE THE CRT CONTROLLER PARAMETERS$'
END
```

A.3 Command Summary

Code	Name	Parameters	Purpose
0	INIT		Initialize the MPIX system.
1	DEFX1	x1	Define the X coordinate of the starting point of the current vector.
2	DEFX2	x2	Define the X coordinate of the terminal point of the current vector.
3	DEFY1	y1	Define the Y coordinate of the starting point of the current vector.
4	DEFY2	y2	Define the Y coordinate of the terminal point of the current vector.
5	DFPT1	x1,y1	Define both coordinates of the starting point of the current vector.
6	DFPT2	x2,y2	Define both coordinates of the terminal point of the current vector.
7	DEFVEC	x1,y1,x2,y2	Define both endpoints of the current vector.
8	WINDX	x1,x2	Define the clipping window in the X direction.
9	WINDY	y1,y2	Define the clipping window in the Y direction.
A	DRAWL		Draw current vector without clipping.
B	DRAWCL		Draw current vector after clipping.
C	SCALX	xfactor	Define scaling factor in X direction .
D	SCALY	yfactor	Define scaling factor in Y direction.
E	SHFTX	xfactor	Define shifting factor in the X direction.
F	SHFTY	yfactor	Define shifting factor in the Y direction.
10	CHAR	ch	Plot a single character at the current cursor position.
11	CHARS	ch,ch..ch,trm	Plot a string of characters.
12	CHREV		Reverse the character mode flag.
13	DEFUS	cnum,cdef	Define a new character.
14	SCREV		Reverse the field mode flag.
15	HOME		Begin new character page.
16	CHPC	x1,y1	Change the current cursor position.
17	CLER		Clear the screen to the value of the field mode flag.
18	CLERR	region	Clear the given region to the value of the field mode flag.
19	COMP		Complement the entire screen.
1A	COMPR	region	Complement the given region of the screen.
1B	CRTAJ	reg,rval	Set the CRTC register reg to rval.

x1,y1,x2,y2	Screen Coordinates. 00 -> 7FH.
xfactor,yfactor	Transformation factors. 8 bit 2's complement number.
ch	A character code. Standard chars. 00 -> 3Fd. User defined characters 80H -> 9BH.
trm	Terminator character. 00H.
cnum	User defined character number. 00 -> 1BH.
cdef	8 bytes which define a user character bit pattern.
region	Two x,y pairs which give the upper left and lower right corners of a region of the screen.
reg	A CRTC register number. 00 -> 0FH.
rval	Value to be placed in the CRTC. 8 bit unsigned.

A.4 System Parameters and Timings

Supply Voltage:	Single 5 volt DC Supply.
Supply Current:	1.7 Amps.
Processor:	Intel 8085.
Host Interface:	Standard S100 Bus. Uses the D00-D03, DI0-DI7, A0-A7, SINP, SOUT, PWR, and PDBIN signals from the bus.
Port Addressing:	Selectable via switch to any group of three contiguous ports which begin at at an address which is a multiple of 3.
System Clock:	Fully tested at 5.7MHz. Designed to operate at full 8085 speed (6 MHz).
Screen Resolution:	128 x 128 points. Individually addressable.
Character Capasity:	Full ASCII character set. Maximum of 12 lines of 20 characters each.
Output:	Standard composite video or, through a converter, standard television carrier.

Timings:

Screen Clear:	50.1 msec.	
Screen Complement	32.8 msec.	
Screen Reverse:	50.0 micsec.	
Home:	170.0 micsec.	
Lines:		
(0,0) -> (7FH,7FH)	21.9 msec	(unclipped)
	27.8 msec	(clipped)
(0,0) -> (40H,40H)	21.7 msec	(unclipped)

A.5 Cost Summary

PART NUMBER	ITEM	QUANTITY	PRICE EA.	TOT. PRICE
-	8 Position DIP Switch	1	2.00	2.00
MC6845	CRT Controller	1	35.00	35.00
2114	Static RAM (14)	4	9.50	38.00
2513	Character Generator	2	10.00	20.00
2716	Erasable ROM (2k x 8)	1	38.00	38.00
-	32 Pin Connector (pair)	1	6.50	6.50
-	40 Pin Connector (pair)	1	8.50	8.50
74LS02	Quad NOR	2	.29	.58
74LS04	Hex Inverter	2	.29	.58
74LS08	Quad NAND	1	.29	.29
74LS138	3 to 8 Decoder	2	2.50	5.00
74LS14	Hex Schmitt Inverters	1	1.70	1.70
74LS157	Quad 2 to 1 Data Sel.	3	1.50	4.50
74LS20	4 Input NAND	1	.29	.29
74LS245	8 bit Bus Driver	2	2.50	5.00
74LS266	Quad EX-NOR	2	.68	1.36
74LS279	Quad SR Latch	1	.79	.79
74LS298	Quad 2 Input Mux.	2	2.00	4.00
74LS32	Quad OR Gates	3	.39	1.17
74LS365	4 bit Tri-state Drivers	4	.87	3.48
74LS74	Dual D Flip-Flop	3	.49	1.47
74S124	Dual VCO	1	2.35	2.35
74S86	Quad EXOR	1	1.10	1.10
74122	One Shot	1	.45	.45
74165	8 bit Shift Register	1	1.40	1.40
8085	8 bit CPU (6 MHz)	1	23.00	23.00
8155	RAM, I/O, Timer	1	27.00	27.00
8212	8 bit I/O Port	3	2.90	8.70
-	Wire and Misc.	-		9.00
-	Vector Board	1	7.50	7.50
-	Wire Wrap Socs. (40 Pin)	3	1.50	4.50
-	Wire Wrap Socs. (24 Pin)	5	.90	4.50
-	Wire Wrap Socs. (20 Pin)	2	.80	1.60
-	Wire Wrap Socs. (18 Pin)	4	.63	2.52
-	Wire Wrap Socs. (16 Pin)	15	.43	6.63
-	Wire Wrap Socs. (14 Pin)	17	.39	6.63
			Grand Total	$281.64

A.6 Wirelist for MPIX

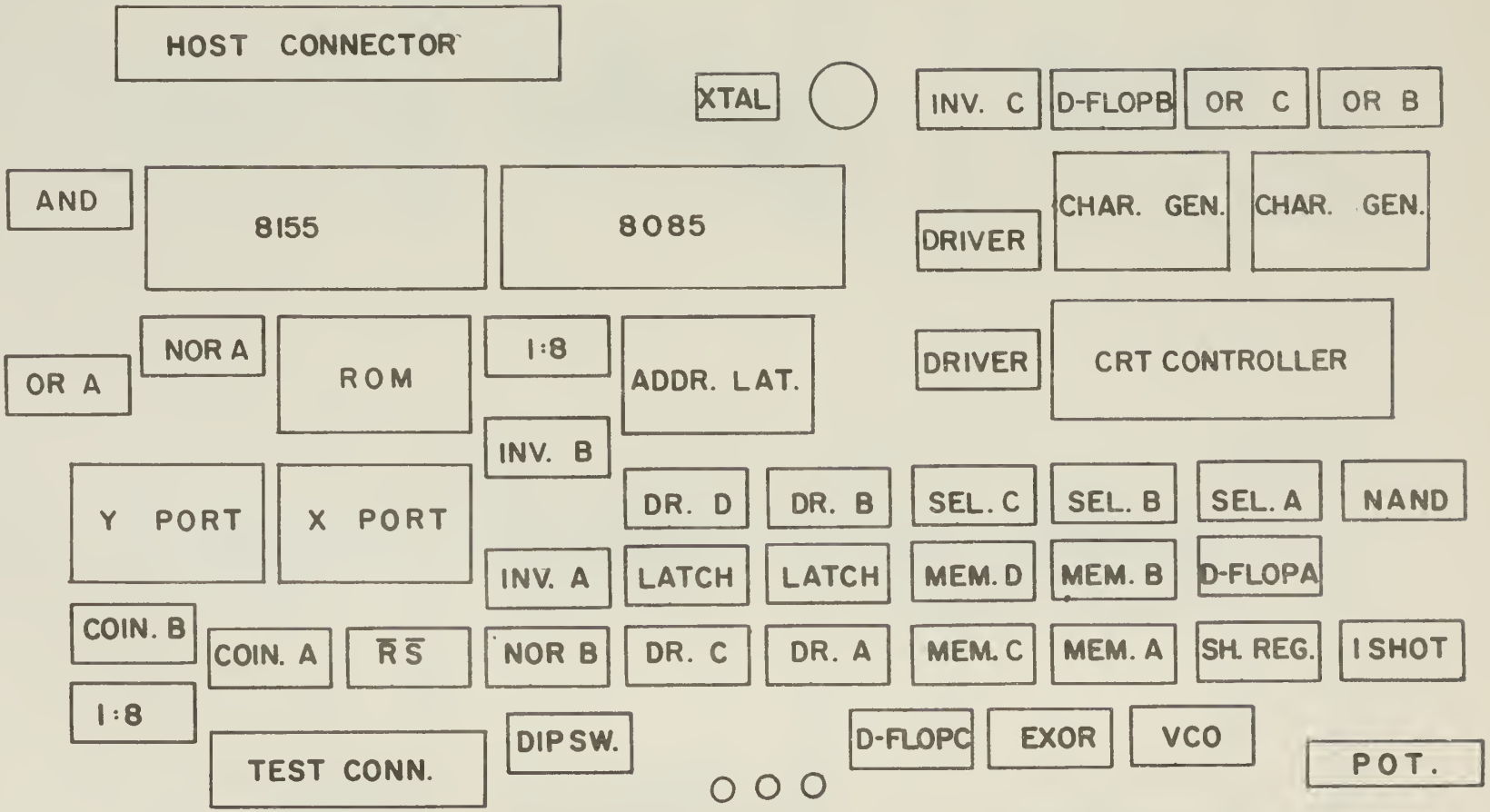
```
********************** WIRELIST - VERSION 1.0 **********************
 THE NUMBER OF CHIPS IS  50.
 THE NUMBER OF DISTINCT LABELS IS 205.

------------------------------          ------------------------------
INTEL PROCESSOR                         UTILITY RAM AND I/O
8C85                                    8155
40 PIN                                  40 PIN
------------------------------          ------------------------------
 1 XTAL          40 5V                   1 NC            40 5V
 2 XTAL          39 GROUND               2 NC            39 (COMSTB)
 3 RESET         38 NC                   3 NC            38 NC
 4 NC            37 SYSCLK               4 RESET         37 NC
 5 GROUND        36 (RST-IN)             5 (DATASTB)     36 HDD7
 6 GROUND        35 READY                6 NC            35 HDD6
 7 GROUND        34 NC                   7 PORTS         34 HDD5
 8 GROUND        33 CPUST1               8 (CERAM)       33 HDD4
 9 GROUND        32 (RD)                 9 (RD)          32 HDD3
10 GROUND        31 (WR)                10 (WR)          31 HDD2
11 NC            30 ALE                 11 ALE           30 HDD1
12 AD0           29 NC                  12 AD0           29 HDD0
13 AD1           28 A15                 13 AD1           28 HDD7
14 AD2           27 A14                 14 AD2           27 HDD6
15 AD3           26 A13                 15 AD3           26 HDD5
16 AD4           25 A12                 16 AD4           25 HDD4
17 AD5           24 A11                 17 AD5           24 HDD3
18 AD6           23 A10                 18 AD6           23 HDD2
19 AD7           22 A09                 19 AD7           22 HDD1
20 GROUND        21 A08                 20 GROUND        21 HDD0

------------------------------          ------------------------------
ADDR BUS DEMUX                          DEVICE SEL DECODER
8212                                    74LS138
24 PIN                                  16 PIN
------------------------------          ------------------------------
 1 GROUND        24 5V                   1 A11           16 5V
 2 GROUND        23 NC                   2 A12           15 (ROMS1)
 3 AD0           22 AD7                  3 A13           14 (RAMS)
 4 A00           21 A07                  4 A15           13 (YSEL)
 5 AD1           20 AD6                  5 GROUND        12 (XSEL)
 6 A01           19 A06                  6 5V            11 (STATS)
 7 AD2           18 AD5                  7 (CRTS)        10 (PORTS)
 8 A02           17 A05                  8 GROUND         9 (CSEMS)
 9 AD3           16 AD4
10 A03           15 A04
11 ALE           14 5V
12 GROUND        13 5V
```

Figure 8

Prototype Layout



PROGRAM ROM
2716
24 PIN

1	A07	24	5V
2	A06	23	A08
3	A05	22	A09
4	A04	21	5V
5	A03	20	(ROMS)
6	A02	19	A10
7	A01	18	GROUND
8	A00	17	D7
9	D0	16	D6
10	D1	15	D5
11	D2	14	D4
12	GROUND	13	D3

AND GATES (A)
74LS08
14 PIN

1	(PORTS)	14	5V
2	(RAMS)	13	NC
3	(CERAM)	12	NC
4	(RD)	11	NC
5	(WR)	10	(CRWAIT)
6	(RDWR)	9	(CSWAIT)
7	GROUND	8	READY

INVERTERS (B)
74LS04
14 PIN

1	(PORTS)	14	5V
2	PORTS	13	ROMS
3	(XSEL)	12	(ROMS)
4	XSEL	11	A06
5	(YSEL)	10	(A06)
6	YSEL	9	HRD
7	GROUND	8	(HRD)

NOR GATES (A)
74LS02
14 PIN

1	ROMS	14	5V
2	(RD)	13	NC
3	(ROMS1)	12	NC
4	(HINOUT)	11	NC
5	SINP	10	NC
6	SOUT	9	NC
7	GROUND	8	NC

DATA SEL. LATCH (U)
74LS298
16 PIN

1	HDO1	16	5V
2	HDO0	15	QA1
3	D0	14	QB1
4	D1	13	QC1
5	HDO2	12	QD1
6	HDO3	11	CLKI
7	D3	10	(SEL)
8	GROUND	9	D2

DATA SEL. LATCH (L)
74LS298
16 PIN

1	NC	16	5V
2	NC	15	NOR-RSV
3	D4	14	QB2
4	D5	13	QC2
5	NC	12	QD2
6	NC	11	CLKV
7	D7	10	(SEL)
8	GROUND	9	D6

BUS DRIVER (B)
74LS365
16 PIN

1	(RD)	16	5V
2	QA1	15	(STATS)
3	D0	14	NC
4	QB1	13	NC
5	D1	12	NC
6	QC1	11	NC
7	D2	10	QD1
8	GROUND	9	D3

BUS DRIVER (D)
74LS365
16 PIN

1	(RD)	16	5V
2	QA2	15	(STATS)
3	D4	14	NC
4	QB2	13	NC
5	D5	12	NC
6	QC2	11	NC
7	D6	10	QD2
8	GROUND	9	D7

BUS DRIVER (A)
74LS365
16 PIN

1	(HSEL)	16	5V
2	NC	15	(HRD)
3	NC	14	QA1
4	NC	13	HDI0
5	NC	12	QB1
6	QD1	11	HDI1
7	HDI3	10	QC1
8	GROUND	9	HDI2

BUS DRIVER (C)
74LS365
16 PIN

1	(HSEL)	16	5V
2	NC	15	(HRD)
3	NC	14	QA2
4	NC	13	NC
5	NC	12	QB2
6	QD2	11	NC
7	NC	10	QC2
8	GROUND	9	NC

(B) (S) LATCHES
74LS279
16 PIN

1	NC	16	5V
2	NC	15	NC
3	NC	14	NC
4	NC	13	NC
5	(ALE)	12	NC
6	(WRS1)	11	NC
7	SEL	10	NC
8	GROUND	9	NC

INVERTERS (A)
74LS04
14 PIN

1	WRS1	14	5V
2	(WRS1)	13	(CLKV)
3	ALE	12	CLKV
4	(ALE)	11	SEL
5	(CLKI)	10	(SEL)
6	CLKI	9	NC
7	GROUND	8	NC

NOR GATES (B)
74LS02
14 PIN

1	WRS1	14	5V
2	(STATS)	13	(CLK1)
3	(WR)	12	WRSI
4	WRSI	11	WRSI
5	(HWR)	10	(CLKV)
6	(HSEL)	9	WRS1
7	GROUND	8	GROUND

"Y" OUTPUT PORT
8212
24 PIN

1	(WR)	24	5V
2	5V	23	NC
3	D0	22	D7
4	TX0	21	TX7
5	D1	20	D6
6	TX1	19	TX6
7	D2	18	D5
8	TX2	17	TX5
9	D3	16	D4
10	TX3	15	TX4
11	GROUND	14	5V
12	GROUND	13	XSEL

"Y" OUTPUT PORT
8212
24 PIN

1	(WR)	24	5V
2	5V	23	NC
3	D0	22	D7
4	TY0	21	TY7
5	D1	20	D6
6	TY1	19	TY6
7	D2	18	D5
8	TY2	17	TY5
9	D3	16	D4
10	TY3	15	TY4
11	GROUND	14	5V
12	GROUND	13	YSEL

DATA BUS DRIVER
74LS245
20 PIN

1	(WR)	20	5V
2	D0	19	DBEN
3	D1	18	AD0
4	D2	17	AD1
5	D3	16	AD2
6	D4	15	AD3
7	D5	14	AD4
8	D6	13	AD5
9	D7	12	AD6
10	GROUND	11	AD7

INVERTERS (C)
74LS14
14 PIN

1	(CERAM)	14	5V
2	CERAM	13	VSYNC
3	SA10	12	(VSYNC)
4	(SA10)	11	HSYNC
5	(RDWR)	10	(HSYNC)
6	RDWR	9	VIDOUT
7	GROUND	8	(VIDOUT)

OR GATES (C)
74LS32
14 PIN

1	(CRD)	14	5V
2	A06	13	NC
3	(ICEN)	12	NC
4	(CRD)	11	NC
5	(A06)	10	(CRTS)
6	(UCEN)	9	(RDY)
7	GROUND	8	(CRWAIT)

OR GATES (B)
74LS32
14 PIN

1	(RDY)	14	5V
2	(CGENS)	13	(SELSC)
3	(CGWAIT)	12	(WR)
4	(RD)	11	DIR
5	(CGENS)	10	(RDWR)
6	(CRD)	9	CERAM
7	GROUND	8	DBEN

DUAL D FLOPS (B)
74LS74
14 PIN

1	(RDY)	14	5V
2	5V	13	5V
3	AIE	12	JUMP1
4	5V	11	SYSCLK
5	JUMP1	10	5V
6	NC	9	NC
7	GROUND	8	(RDY)

CHARACTER GEN. (U)
2513
24 PIN

1	NC	24	5V
2	NC	23	NC
3	NC	22	A05
4	D4	21	A04
5	D3	20	A03
6	D2	19	A02
7	D1	18	A01
8	D0	17	A00
9	NC	16	A10
10	GROUND	15	A09
11	(UCEN)	14	A08
12	NC	13	NC

CHARACTER GEN. (L)
2513
24 PIN

1	NC	24	5V
2	NC	23	NC
3	NC	22	A05
4	D4	21	A04
5	D3	20	A03
6	D2	19	A02
7	D1	18	A01
8	D0	17	A00
9	NC	16	A10
10	GROUND	15	A09
11	(LCEN)	14	A08
12	NC	13	NC

CRT CONTROLLER
MC6845
40 PIN

1	GROUND	40	VSYNC
2	(RST-IN)	39	HSYNC
3	GROUND	38	NC
4	VA00	37	NC
5	VA01	36	NC
6	VA02	35	NC
7	VA03	34	NC
8	VA04	33	D0
9	VA05	32	D1
10	VA06	31	D2
11	VA07	30	D3
12	VA08	29	D4
13	VA09	28	D5
14	VA10	27	D6
15	VA11	26	D7
16	VA12	25	(CRTS)
17	VA13	24	A00
18	DISPEN	23	RDWR
19	CURSOR	22	CPUST1
20	5V	21	VIDCLK2

SCREEN BUS DRIVER
74LS245
20 PIN

1	DIR	20	5V
2	DS0	19	(SELSC)
3	DS1	18	D0
4	DS2	17	D1
5	DS3	16	D2
6	DS4	15	D3
7	DS5	14	D4
8	DS6	13	D5
9	DS7	12	D6
10	GROUND	11	D7

SCREEN ADD. SEL. (A)
74LS157
16 PIN

1	(SELSC)	16	5V
2	A03	15	GROUND
3	VA02	14	A06
4	SA00	13	VA05
5	A04	12	SA03
6	VA03	11	A05
7	SA01	10	VA04
8	GROUND	9	SA02

SCREEN ADD. SEL. (B)
74LS157
16 PIN

1	(SELSC)	16	5V
2	A08	15	GROUND
3	VA06	14	A11
4	SA04	13	VA09
5	A09	12	SA07
6	VA07	11	A10
7	SA05	10	VA08
8	GROUND	9	SA06

SCREEN ADD. SEL. (C)
74LS157
16 PIN

1	(SELSC)	16	5V
2	A12	15	GROUND
3	VA10	14	NC
4	SA08	13	NC
5	A13	12	NC
6	VA11	11	A14
7	SA09	10	VA12
8	GROUND	9	SA10

DUAL D FLOP (A)
74LS74
14 PIN

1	5V	14	5V
2	A15	13	5V
3	(ALE)	12	VIDCLK2
4	5V	11	VIDCLK
5	NC	10	5V
6	(SELSC)	9	NC
7	GROUND	8	VIDCLK2

SCREEN RAM (A)
2114
18 PIN

1	SA06	18	5V
2	SA05	17	SA07
3	SA04	16	SA08
4	SA03	15	SA09
5	SA00	14	DS0
6	SA01	13	DS1
7	SA02	12	DS2
8	SA10	11	DS3
9	GROUND	10	DIR

SCREEN RAM (B)
2114
18 PIN

1	SA06	18	5V
2	SA05	17	SA07
3	SA04	16	SA08
4	SA03	15	SA09
5	SA00	14	DS0
6	SA01	13	DS1
7	SA02	12	DS2
8	(SA10)	11	DS3
9	GROUND	10	DIR

SCREEN RAM (C)
2114
18 PIN

1	SA06	18	5V
2	SA05	17	SA07
3	SA04	16	SA08
4	SA03	15	SA09
5	SA00	14	DS4
6	SA01	13	DS5
7	SA02	12	DS6
8	SA10	11	DS7
9	GROUND	10	DIR

SCREEN RAM (D)
2114
18 PIN

1	SA06	18	5V
2	SA05	17	SA07
3	SA04	16	SA08
4	SA03	15	SA09
5	SA00	14	DS4
6	SA01	13	DS5
7	SA02	12	DS6
8	(SA10)	11	DS7
9	GROUND	10	DIR

VIDEO SHIFT RLGISTER
74165
16 PIN

1	(PLOAD)	16	5V
2	VIDCLK	15	GROUND
3	DS3	14	DS4
4	DS2	13	DS5
5	DS1	12	DS6
6	DS0	11	DS7
7	NC	10	NC
8	GROUND	9	VIDEO

NAND GATES (4 INPUT)
74LS20
14 PIN

1	VIDCLK2	14	5V
2	VA00	13	NC
3	NC	12	NC
4	5V	11	NC
5	VA01	10	NC
6	(LOAD)	9	NC
7	GROUND	8	NC

DUAL VCO (3 MHZ)
74S124
16 PIN

1	NC	16	5V
2	POT	15	5V
3	5V	14	NC
4	C130PF	13	NC
5	C130PF	12	NC
6	GROUND	11	NC
7	VIDCLK	10	NC
8	GROUND	9	GROUND

ONE SHOT (100 NSEC)
74122
14 PIN

1	(LOAD)	14	5V
2	5V	13	RCEXT
3	5V	12	NC
4	5V	11	C27PF
5	5V	10	NC
6	(PLOAD)	9	NC
7	GROUND	8	NC

EXCL. OR-NOR GATES
74S86
14 PIN

1	VIDEO	14	5V
2	NOR-REV	13	NC
3	JUMP2	12	NC
4	JUMP2	11	NC
5	5V	10	CURSOR
6	VIDO1	9	VIDO1
7	GROUND	8	VIDO2

3 TO 8 DECODER
74LS138
16 PIN

1	HA0	16	5V
2	HA1	15	(HSEL)
3	GROUND	14	(DSTB3)
4	GROUND	13	(CSTB3)
5	(HINOUT)	12	NC
6	AID=	11	NC
7	NC	10	NC
8	GROUND	9	NC

COINCIDENCE GATE (A)
74LS266
14 PIN

1	HA2	14	5V
2	ADDS2	13	ADDS5
3	ADD=	12	HA5
4	ADD=	11	ADD=
5	HA3	10	ADD=
6	ADDS3	9	ADDS4
7	GROUND	8	HA4

COINCIDENCE GATE (B)
74LS266
14 PIN

1	HA6	14	5V
2	ADDS6	13	NC
3	ADD=	12	NC
4	ALD=	11	NC
5	HA7	10	NC
6	ADDS7	9	NC
7	GROUND	8	NC

OR GATES (A)
74LS32
14 PIN

1	(HWR)	14	5V
2	(CSTBE)	13	(HSEL)
3	(COMSTB)	12	(HRD)
4	(HWR)	11	(BUFFEN)
5	(DSTBE)	10	(VIDEN)
6	(DATASTB)	9	VIDO3
7	GROUND	8	VIDOUT

HOST CONNECTOR
40P CON.
40 PIN

1	HRD	40	GROUND
2	(HWR)	39	5V
3	SINP	38	(BUFFEN)
4	SOUT	37	NC
5	HA7	36	NC
6	HA6	35	NC
7	HA5	34	NC
8	HA4	33	NC
9	HA3	32	NC
10	HA2	31	NC
11	HA1	30	NC
12	HA0	29	NC
13	HDO7	28	HDI7
14	HDO6	27	HDI6
15	HDO5	26	HDI5
16	HDO4	25	HDI4
17	HDO3	24	HDI3
18	HDO2	23	HDI2
19	HDO1	22	HDI1
20	HDO0	21	HDI0

PORT ADDR. SELECT
DIP SW.
16 PIN

1	NC	16	5V
2	NC	15	5V
3	ADDS2	14	5V
4	ADDS3	13	5V
5	ADDS4	12	5V
6	ADDS5	11	5V
7	ADDS6	10	5V
8	ADDS7	9	5V

DUAL D FLOP (C)
74LS74
14 PIN

1	5V	14	5V
2	DISPEN	13	NC
3	(LOAD)	12	VIDO2
4	5V	11	VIDCLK
5	NC	10	NC
6	(VIDEN)	9	VIDO3
7	GROUND	8	NC

TEST CONNECTOR
32P CON.
34 PIN

1	GROUND	34	5V
2	NC	33	SYSCLK
3	TY6	32	TY7
4	TY4	31	TY5
5	TY2	30	TY3
6	TY0	29	TY1
7	TX6	28	TX7
8	TX4	27	TX5
9	TX2	26	TX3
10	TX0	25	TX1
11	NC	24	(RST-IN)
12	NC	23	NC
13	NC	22	NC
14	NC	21	NC
15	NC	20	NC
16	NC	19	NC
17	NC	18	NC

********** CROSS REFERENCE LISTING **********

ALDS2	COINCIDENCE GATE (A)	74LS266	2
ADDS2	PORT ADDR. SELECT	DIP SW.	3
ADDS3	COINCIDENCE GATE (A)	74LS266	6
ADDS3	PORT ADDR. SELECT	DIP SW.	4
ADDS4	COINCIDENCE GATE (A)	74LS266	9
ADDS4	PORT ADDR. SELECT	DIP SW.	5
ADDS5	COINCIDENCE GATE (A)	74LS266	13
ADDS5	PORT ADDR. SELECT	DIP SW.	6
ADDS6	COINCIDENCE GATE (B)	74LS266	2
ADDS6	PORT ADDR. SELECT	DIP SW.	7
ADDS7	COINCIDENCE GATE (B)	74LS266	6
ADDS7	PORT ADDR. SELECT	DIP SW.	8
ADD=	3 TO 8 DECODER	74LS138	6
ADD=	COINCIDENCE GATE (A)	74LS266	3
ADD=	COINCIDENCE GATE (A)	74LS266	4
ADD=	COINCIDENCE GATE (A)	74LS266	10
ADD=	COINCIDENCE GATE (A)	74LS266	11
ALD=	COINCIDENCE GATE (B)	74LS266	3
ADD=	COINCIDENCE GATE (B)	74LS266	4
AD0	INTEL PROCESSOR	8085	12
AD0	UTILITY RAM AND I/O	8155	12
AD0	ADDR BUS DEMUX	8212	3
AD0	DATA BUS DRIVER	74LS245	18
AD1	INTEL PROCESSOR	8085	13
AD1	UTILITY RAM AND I/O	8155	13
AD1	ADDR BUS DEMUX	8212	5
AD1	DATA BUS DRIVER	74LS245	17
AD2	INTEL PROCESSOR	8085	14
AD2	UTILITY RAM AND I/O	8155	14
AD2	ADDR BUS DEMUX	8212	7
AD2	DATA BUS DRIVER	74LS245	15
AD3	INTEL PROCESSOR	8085	15
AD3	UTILITY RAM AND I/O	8155	15
AD3	ADDR BUS DEMUX	8212	9
AD3	DATA BUS DRIVER	74LS245	15
AD4	INTEL PROCESSOR	8085	16
AD4	UTILITY RAM AND I/O	8155	16
AD4	ADDR BUS DEMUX	8212	16
AD4	DATA BUS DRIVER	74LS245	14

AD5	INTEL PROCESSOR	8085	17
AD5	UTILITY RAM AND I/O	8155	17
AD5	ADDR BUS DEMUX	8212	18
AD5	DATA BUS DRIVER	74LS245	13
AD6	INTEL PROCESSOR	8085	18
AD6	UTILITY RAM AND I/O	8155	18
AD6	ADDR BUS DEMUX	8212	20
AD6	DATA BUS DRIVER	74LS245	12
AD7	INTEL PROCESSOR	8085	19
AD7	UTILITY RAM AND I/O	8155	19
AD7	ADDR BUS DEMUX	8212	22
AD7	DATA BUS DRIVER	74LS245	11
ALE	INTEL PROCESSOR	8085	30
ALE	UTILITY RAM AND I/O	8155	11
ALE	ADDR BUS DEMUX	8212	11
ALE	INVERTERS (A)	74LS04	3
ALE	DUAL D FLOPS (B)	74LS74	3
A00	ADDR BUS DEMUX	8212	4
A00	PROGRAM ROM	2716	8
A00	CHARACTER GEN. (U)	2513	17
A00	CHARACTER GEN. (L)	2513	17
A00	CRT CONTROLLER	MC6845	24
A01	ADDR BUS DEMUX	8212	6
A01	PROGRAM ROM	2716	7
A01	CHARACTER GEN. (U)	2513	18
A01	CHARACTER GEN. (L)	2513	18
A02	ADDR BUS DEMUX	8212	8
A02	PROGRAM ROM	2716	6
A02	CHARACTER GEN. (U)	2513	19
A02	CHARACTER GEN. (L)	2513	19
A03	ADDR BUS DEMUX	8212	10
A03	PROGRAM ROM	2716	5
A03	CHARACTER GEN. (U)	2513	20
A03	CHARACTER GEN. (L)	2513	20
A03	SCREEN ADD. SEL. (A)	74LS157	2
A04	ADDR BUS DEMUX	8212	15
A04	PROGRAM ROM	2716	4
A04	CHARACTER GEN. (U)	2513	21
A04	CHARACTER GEN. (L)	2513	21
A04	SCREEN ADD. SEL. (A)	74LS157	5
A05	ADDR BUS DEMUX	8212	17
A05	PROGRAM ROM	2716	3
A05	CHARACTER GEN. (U)	2513	22

A05	CHARACTER GEN. (L)	2513	22
A05	SCREEN ADD. SEL. (A)	74LS157	11
A06	ADDR BUS DEMUX	8212	19
A06	PROGRAM ROM	2716	2
A06	INVERTERS (B)	74LS04	11
A06	OR GATES (C)	74LS32	2
A06	SCREEN ADD. SEL. (A)	74LS157	14
A07	ADDR BUS DEMUX	8212	21
A07	PROGRAM ROM	2716	1
A08	INTEL PROCESSOR	8085	21
A08	PROGRAM ROM	2716	23
A08	CHARACTER GEN. (U)	2513	14
A08	CHARACTER GEN. (L)	2513	14
A08	SCREEN ADD. SEL. (B)	74LS157	2
A09	INTEL PROCESSOR	8085	22
A09	PROGRAM ROM	2716	22
A09	CHARACTER GEN. (U)	2513	15
A09	CHARACTER GEN. (L)	2513	15
A09	SCREEN ADD. SEL. (B)	74LS157	5
A10	INTEL PROCESSOR	8085	23
A10	PROGRAM ROM	2716	19
A10	CHARACTER GEN. (U)	2513	16
A10	CHARACTER GEN. (L)	2513	16
A10	SCREEN ADD. SEL. (B)	74LS157	11
A11	INTEL PROCESSOR	8085	24
A11	DEVICE SEL DECODER	74LS138	1
A11	SCREEN ADD. SEL. (B)	74LS157	14
A12	INTEL PROCESSOR	8085	25
A12	DEVICE SEL DECODER	74LS138	2
A12	SCREEN ADD. SEL. (C)	74LS157	2
A13	INTEL PROCESSOR	8085	26
A13	DEVICE SEL DECODER	74LS138	3
A13	SCREEN ADD. SEL. (C)	74LS157	5
A14	INTEL PROCESSOR	8085	27
A14	SCREEN ADD. SEL. (C)	74LS157	11
A15	INTEL PROCESSOR	8085	28
A15	DEVICE SEL DECODER	74LS138	4
A15	DUAL D FLOP (A)	74LS74	2
CERAM	INVERTERS (C)	74LS14	2
CERAM	OR GATES (B)	74LS32	9
CIKI	DATA SEL. LATCH (U)	74LS298	11

CLKI	INVERTERS (A)	74LS04	6
CLKV	DATA SEL. LATCH (L)	74LS298	11
CLKV	INVERTERS (A)	74LS04	12
CPUST1	INTEL PROCESSOR	8085	33
CPUST1	CRT CONTROLLER	MC6845	22
CURSOR	CRT CONTROLLER	MC6845	19
CURSOR	EXCL. OR-NOR GATES	74S86	10
C130PF	DUAL VCO (3 MHZ)	74S124	4
C130PF	DUAL VCO (3 MHZ)	74S124	5
C27PF	ONE SHOT (100 NSEC)	74122	11
DBEN	DATA BUS DRIVER	74LS245	19
DBEN	OR GATES (B)	74LS32	8
DIR	OR GATES (B)	74LS32	11
DIR	SCREEN BUS DRIVER	74LS245	1
DIR	SCREEN RAM (A)	2114	10
DIR	SCREEN RAM (B)	2114	10
DIR	SCREEN RAM (C)	2114	10
DIR	SCREEN RAM (D)	2114	10
DISPEN	CRT CONTROLLER	MC6845	18
DISPEN	DUAL D FLOP (C)	74LS74	2
DS0	SCREEN BUS DRIVER	74LS245	2
DS0	SCREEN RAM (A)	2114	14
DS0	SCREEN RAM (B)	2114	14
DS0	VIDEO SHIFT REGISTER	74165	6
DS1	SCREEN BUS DRIVER	74LS245	3
DS1	SCREEN RAM (A)	2114	13
DS1	SCREEN RAM (B)	2114	13
DS1	VIDEO SHIFT REGISTER	74165	5
DS2	SCREEN BUS DRIVER	74LS245	4
DS2	SCREEN RAM (A)	2114	12
DS2	SCREEN RAM (B)	2114	12
DS2	VIDEO SHIFT REGISTER	74165	4
DS3	SCREEN BUS DRIVER	74LS245	5
DS3	SCREEN RAM (A)	2114	11
DS3	SCREEN RAM (B)	2114	11
DS3	VIDEO SHIFT REGISTER	74165	3
DS4	SCREEN BUS DRIVER	74LS245	6
DS4	SCREEN RAM (C)	2114	14
DS4	SCREEN RAM (D)	2114	14
DS4	VIDEO SHIFT REGISTER	74165	14

DS5	SCREEN BUS DRIVER	74LS245	7
DS5	SCREEN RAM (C)	2114	13
DS5	SCREEN RAM (D)	2114	13
DS5	VIDEO SHIFT REGISTER	74165	13
DS6	SCREEN BUS DRIVER	74LS245	8
DS6	SCREEN RAM (C)	2114	12
DS6	SCREEN RAM (D)	2114	12
DS6	VIDEO SHIFT REGISTER	74165	12
DS7	SCREEN BUS DRIVER	74LS245	9
DS7	SCREEN RAM (C)	2114	11
DS7	SCREEN RAM (D)	2114	11
DS7	VIDEO SHIFT REGISTER	74165	11
D0	PROGRAM ROM	2716	9
D0	DATA SEL. LATCH (U)	74LS298	3
D0	BUS DRIVER (B)	74LS365	3
D0	"X" OUTPUT PORT	8212	3
D0	"Y" OUTPUT PORT	8212	3
D0	DATA BUS DRIVER	74LS245	2
D0	CHARACTER GEN. (U)	2513	8
D0	CHARACTER GEN. (L)	2513	8
D0	CRT CONTROLLER	MC6845	33
D0	SCREEN BUS DRIVER	74LS245	18
D1	PROGRAM ROM	2716	10
D1	DATA SEL. LATCH (U)	74LS298	4
D1	BUS DRIVER (B)	74LS365	5
D1	"X" OUTPUT PORT	8212	5
D1	"Y" OUTPUT PORT	8212	5
D1	DATA BUS DRIVER	74LS245	3
D1	CHARACTER GEN. (U)	2513	7
D1	CHARACTER GEN. (L)	2513	7
D1	CRT CONTROLLER	MC6845	32
D1	SCREEN BUS DRIVER	74LS245	17
D2	PROGRAM ROM	2716	11
D2	DATA SEL. LATCH (U)	74LS298	9
D2	BUS DRIVER (B)	74LS365	7
D2	"X" OUTPUT PORT	8212	7
D2	"Y" OUTPUT PORT	8212	7
D2	DATA BUS DRIVER	74LS245	4
D2	CHARACTER GEN. (U)	2513	6
D2	CHARACTER GEN. (L)	2513	6
D2	CRT CONTROLLER	MC6845	31
D2	SCREEN BUS DRIVER	74LS245	16
D3	PROGRAM ROM	2716	13
D3	DATA SEL. LATCH (U)	74LS298	7
D3	BUS DRIVER (B)	74LS365	9
D3	"X" OUTPUT PORT	8212	9

D3	"Y" OUTPUT PORT	8212	9
D3	DATA BUS DRIVER	74LS245	5
D3	CHARACTER GEN. (U)	2513	5
D3	CHARACTER GEN. (L)	2513	5
D3	CRT CONTROLLER	MC6845	30
D3	SCREEN BUS DRIVER	74LS245	15
D4	PROGRAM ROM	2716	14
D4	DATA SEL. LATCH (L)	74LS298	3
D4	BUS DRIVER (D)	74LS365	3
D4	"X" OUTPUT PORT	8212	16
D4	"Y" OUTPUT PORT	8212	16
D4	DATA BUS DRIVER	74LS245	6
D4	CHARACTER GEN. (U)	2513	4
D4	CHARACTER GEN. (L)	2513	4
D4	CRT CONTROLLER	MC6845	29
D4	SCREEN BUS DRIVER	74LS245	14
D5	PROGRAM ROM	2716	15
D5	DATA SEL. LATCH (L)	74LS298	4
D5	BUS DRIVER (D)	74LS365	5
D5	"X" OUTPUT PORT	8212	18
D5	"Y" OUTPUT PORT	8212	18
D5	DATA BUS DRIVER	74LS245	7
D5	CRT CONTROLLER	MC6845	28
D5	SCREEN BUS DRIVER	74LS245	13
D6	PROGRAM ROM	2716	16
D6	DATA SEL. LATCH (L)	74LS298	9
D6	BUS DRIVER (D)	74LS365	7
D6	"X" OUTPUT PORT	8212	20
D6	"Y" OUTPUT PORT	8212	20
D6	DATA BUS DRIVER	74LS245	8
D6	CRT CONTROLLER	MC6845	27
D6	SCREEN BUS DRIVER	74LS245	12
D7	PROGRAM ROM	2716	17
D7	DATA SEL. LATCH (L)	74LS298	7
D7	BUS DRIVER (D)	74LS365	9
D7	"X" OUTPUT PORT	8212	22
D7	"Y" OUTPUT PORT	8212	22
D7	DATA BUS DRIVER	74LS245	9
D7	CRT CONTROLLER	MC6845	26
D7	SCREEN BUS DRIVER	74LS245	11
GROUND	INTEL PROCESSOR	8085	5
GROUND	INTEL PROCESSOR	8085	6
GROUND	INTEL PROCESSOR	8085	7
GROUND	INTEL PROCESSOR	8085	8
GROUND	INTEL PROCESSOR	8085	9
GROUND	INTEL PROCESSOR	8085	10
GROUND	INTEL PROCESSOR	8085	20
GROUND	INTEL PROCESSOR	8085	39

GROUND	UTILITY RAM AND I/O	8155	20
GROUND	ADDR BUS DEMUX	8212	1
GROUND	ADDR BUS DEMUX	8212	2
GROUND	ADDR BUS DEMUX	8212	12
GROUND	DEVICE SEL DECODER	74LS138	5
GROUND	DEVICE SEL DECODER	74LS138	8
GROUND	PROGRAM ROM	2716	12
GROUND	PROGRAM ROM	2716	13
GROUND	AND GATES (A)	74LS08	7
GROUND	INVERTERS (B)	74LS04	7
GROUND	NOR GATES (A)	74LS02	7
GROUND	DATA SEL. LATCH (U)	74LS298	8
GROUND	DATA SEL. LATCH (L)	74LS298	8
GROUND	BUS DRIVER (B)	74LS365	8
GROUND	BUS DRIVER (D)	74LS365	8
GROUND	BUS DRIVER (A)	74LS365	8
GROUND	BUS DRIVER (C)	74LS365	8
GROUND	(R) (S) LATCHES	74LS279	8
GROUND	INVERTERS (A)	74LS04	7
GROUND	NOR GATES (B)	74LS02	7
GROUND	NOR GATES (B)	74LS02	8
GROUND	"Y" OUTPUT PORT	8212	11
GROUND	"X" OUTPUT PORT	8212	12
GROUND	"Y" OUTPUT PORT	8212	11
GROUND	"Y" OUTPUT PORT	8212	12
GROUND	DATA BUS DRIVER	74LS245	10
GROUND	INVERTERS (C)	74LS14	7
GROUND	OR GATES (C)	74LS32	7
GROUND	OR GATES (B)	74LS32	7
GROUND	DUAL D FLOPS (B)	74LS74	7
GROUND	CHARACTER GEN. (U)	2513	10
GROUND	CHARACTER GEN. (L)	2513	10
GROUND	CRT CONTROLLER	MC6845	1
GROUND	CRT CONTROLLER	MC6845	3
GROUND	SCREEN BUS DRIVER	74LS245	10
GROUND	SCREEN ADD. SEL. (A)	74LS157	8
GROUND	SCREEN ADD. SEL. (A)	74LS157	15
GROUND	SCREEN ADD. SEL. (B)	74LS157	8
GROUND	SCREEN ADD. SEL. (B)	74LS157	15
GROUND	SCREEN ADD. SEL. (C)	74LS157	8
GROUND	SCREEN ADD. SEL. (C)	74LS157	15
GROUND	DUAL D FLOP (A)	74LS74	7
GROUND	SCREEN RAM (A)	2114	9
GROUND	SCREEN RAM (B)	2114	9
GROUND	SCREEN RAM (C)	2114	9
GROUND	SCREEN RAM (D)	2114	9
GROUND	VIDEO SHIFT REGISTER	74165	8
GROUND	VIDEO SHIFT REGISTER	74165	15
GROUND	NAND GATES (4 INPUT)	74LS20	7
GROUND	DUAL VCO (3 MHZ)	74S124	5
GROUND	DUAL VCO (3 MHZ)	74S124	8
GROUND	DUAL VCO (3 MHZ)	74S124	9
GROUND	ONE SHOT (100 NSEC)	74122	7

GROUND	EXCL. OR-NOR GATES	74S86	7
GROUND	3 TO 8 DECODER	74LS138	3
GROUND	3 TO 8 DECODER	74LS138	4
GROUND	3 TO 8 DECODER	74LS138	8
GROUND	COINCIDENCE GATE (A)	74LS266	7
GROUND	COINCIDENCE GATE (B)	74LS266	7
GROUND	OR GATES (A)	74LS32	7
GROUND	DUAL D FLOP (C)	74LS74	7
GROUND	HOST CONNECTOR	40P CON.	40
GROUND	TEST CONNECTOR	32P CON.	1
HA0	3 TO 8 DECODER	74LS138	1
HA0	HOST CONNECTOR	40P CON.	12
HA1	3 TO 8 DECODER	74LS138	2
HA1	HOST CONNECTOR	40P CON.	11
HA2	COINCIDENCE GATE (A)	74LS266	1
HA2	HOST CONNECTOR	40P CON.	10
HA3	COINCIDENCE GATE (A)	74LS266	5
HA3	HOST CONNECTOR	40P CON.	9
HA4	COINCIDENCE GATE (A)	74LS266	8
HA4	HOST CONNECTOR	40P CON.	8
HA5	COINCIDENCE GATE (A)	74LS266	12
HA5	HOST CONNECTOR	40P CON.	7
HA6	COINCIDENCE GATE (B)	74LS266	1
HA6	HOST CONNECTOR	40P CON.	6
HA7	COINCIDENCE GATE (B)	74LS266	5
HA7	HOST CONNECTOR	40P CON.	5
HDI0	BUS DRIVER (A)	74LS365	13
HDI0	HOST CONNECTOR	40P CON.	21
HDI1	BUS DRIVER (A)	74LS365	11
HDI1	HOST CONNECTOR	40P CON.	22
HDI2	BUS DRIVER (A)	74LS365	9
HDI2	HOST CONNECTOR	40P CON.	23
HDI3	BUS DRIVER (A)	74LS365	7
HDI3	HOST CONNECTOR	40P CON.	24
HDI4	HOST CONNECTOR	40P CON.	25
HDI5	HOST CONNECTOR	40P CON.	26
HDI6	HOST CONNECTOR	40P CON.	27

HDI7	HOST CONNECTOR	40P CON.	28
HDO0	UTILITY RAM AND I/O	8155	21
HDO0	UTILITY RAM AND I/O	8155	29
HDO0	DATA SEL. LATCH (U)	74LS298	2
HDO0	HOST CONNECTOR	40P CON.	20
HDO1	UTILITY RAM AND I/O	8155	22
HDO1	UTILITY RAM AND I/O	8155	30
HDO1	DATA SEL. LATCH (U)	74LS298	1
HDO1	HOST CONNECTOR	40P CON.	19
HDO2	UTILITY RAM AND I/O	8155	23
HDO2	UTILITY RAM AND I/O	8155	31
HDO2	DATA SEL. LATCH (U)	74LS298	5
HDO2	HOST CONNECTOR	40P CON.	18
HDO3	UTILITY RAM AND I/O	8155	24
HDO3	UTILITY RAM AND I/O	8155	32
HDO3	DATA SEL. LATCH (U)	74LS298	6
HDO3	HOST CONNECTOR	40P CON.	17
HDO4	UTILITY RAM AND I/O	8155	25
HDO4	UTILITY RAM AND I/O	8155	33
HDO4	HOST CONNECTOR	40P CON.	16
HDO5	UTILITY RAM AND I/O	8155	26
HDO5	UTILITY RAM AND I/O	8155	34
HDO5	HOST CONNECTOR	40P CON.	15
HDO6	UTILITY RAM AND I/O	8155	27
HDO6	UTILITY RAM AND I/O	8155	35
HDO6	HOST CONNECTOR	40P CON.	14
HDO7	UTILITY RAM AND I/O	8155	28
HDO7	UTILITY RAM AND I/O	8155	36
HDO7	HOST CONNECTOR	40P CON.	13
HRD	INVERTERS (B)	74LS04	9
HRD	HOST CONNECTOR	40P CON.	1
HSYNC	INVERTERS (C)	74LS14	11
HSYNC	CRT CONTROLLER	MC6845	39
JUMP1	DUAL D FLOPS (B)	74LS74	5
JUMP1	DUAL D FLOPS (B)	74LS74	12
JUMP2	EXCL. OR-NOR GATES	74S86	3
JUMP2	EXCL. OR-NOR GATES	74S86	4
NCR-REV	DATA SEL. LATCH (L)	74LS298	15
NCR-REV	EXCL. OR-NOR GATES	74S86	2

PORTS	UTILITY RAM AND I/O	8155	7
PORTS	INVERTERS (B)	74LS04	2
PCT	DUAL VCO (3 MHZ)	74S124	2
QA1	DATA SEL. LATCH (U)	74LS298	15
QA1	BUS DRIVER (B)	74LS365	2
QA1	BUS DRIVER (A)	74LS365	14
QA2	BUS DRIVER (D)	74LS365	2
QA2	BUS DRIVER (C)	74LS365	14
QB1	DATA SEL. LATCH (U)	74LS298	14
QB1	BUS DRIVER (B)	74LS365	4
QB1	BUS DRIVER (A)	74LS365	12
QB2	DATA SEL. LATCH (L)	74LS298	14
QB2	BUS DRIVER (D)	74LS365	4
QB2	BUS DRIVER (C)	74LS365	12
QC1	DATA SEL. LATCH (U)	74LS298	13
QC1	BUS DRIVER (B)	74LS365	6
QC1	BUS DRIVER (A)	74LS365	10
QC2	DATA SEL. LATCH (L)	74LS298	13
QC2	BUS DRIVER (D)	74LS365	6
QC2	BUS DRIVER (C)	74LS365	10
QD1	DATA SEL. LATCH (U)	74LS298	12
QD1	BUS DRIVER (B)	74LS365	10
QD1	BUS DRIVER (A)	74LS365	6
QD2	DATA SEL. LATCH (L)	74LS298	12
QD2	BUS DRIVER (D)	74LS365	10
QD2	BUS DRIVER (C)	74LS365	6
RCEXT	ONE SHOT (100 NSEC)	74122	13
RDWR	INVERTERS (C)	74LS14	6
RDWR	CRT CONTROLLER	MC6845	23
READY	INTEL PROCESSOR	8085	35
READY	AND GATES (A)	74LS08	8
RESET	INTEL PROCESSOR	8085	3
RESET	UTILITY RAM AND I/O	8155	4
RCMS	INVERTERS (B)	74LS04	13
RCMS	NOR GATES (A)	74LS02	1
SA00	SCREEN ADD. SEL. (A)	74LS157	4
SA00	SCREEN RAM (A)	2114	5
SA00	SCREEN RAM (B)	2114	5

SA00	SCREEN RAM (C)	2114	5
SA00	SCREEN RAM (D)	2114	5
SA01	SCREEN ADD. SEL. (A)	74LS157	7
SA01	SCREEN RAM (A)	2114	6
SA01	SCREEN RAM (B)	2114	6
SA01	SCREEN RAM (C)	2114	6
SA01	SCREEN RAM (D)	2114	6
SA02	SCREEN ADD. SEL. (A)	74LS157	9
SA02	SCREEN RAM (A)	2114	7
SA02	SCREEN RAM (B)	2114	7
SA02	SCREEN RAM (C)	2114	7
SA02	SCREEN RAM (D)	2114	7
SA03	SCREEN ADD. SEL. (A)	74LS157	12
SA03	SCREEN RAM (A)	2114	4
SA03	SCREEN RAM (B)	2114	4
SA03	SCREEN RAM (C)	2114	4
SA03	SCREEN RAM (D)	2114	4
SA04	SCREEN ADD. SEL. (B)	74LS157	4
SA04	SCREEN RAM (A)	2114	3
SA04	SCREEN RAM (B)	2114	3
SA04	SCREEN RAM (C)	2114	3
SA04	SCREEN RAM (D)	2114	3
SA05	SCREEN ADD. SEL. (B)	74LS157	7
SA05	SCREEN RAM (A)	2114	2
SA05	SCREEN RAM (B)	2114	2
SA05	SCREEN RAM (C)	2114	2
SA05	SCREEN RAM (D)	2114	2
SA06	SCREEN ADD. SEL. (B)	74LS157	9
SA06	SCREEN RAM (A)	2114	1
SA06	SCREEN RAM (B)	2114	1
SA06	SCREEN RAM (C)	2114	1
SA06	SCREEN RAM (D)	2114	1
SA07	SCREEN ADD. SEL. (B)	74LS157	12
SA07	SCREEN RAM (A)	2114	17
SA07	SCREEN RAM (B)	2114	17
SA07	SCREEN RAM (C)	2114	17
SA07	SCREEN RAM (D)	2114	17
SA08	SCREEN ADD. SEL. (C)	74LS157	4
SA08	SCREEN RAM (A)	2114	16
SA08	SCREEN RAM (B)	2114	16
SA08	SCREEN RAM (C)	2114	16
SA08	SCREEN RAM (D)	2114	16
SA09	SCREEN ADD. SEL. (C)	74LS157	7
SA09	SCREEN RAM (A)	2114	15

SA09	SCREEN RAM (B)	2114	15
SA09	SCREEN RAM (C)	2114	15
SA09	SCREEN RAM (D)	2114	15
SA10	INVERTERS (C)	74LS14	3
SA10	SCREEN ADD. SEL. (C)	74LS157	9
SA10	SCREEN RAM (A)	2114	8
SA10	SCREEN RAM (C)	2114	8
SEL	(R) (S) LATCHES	74LS279	7
SEL	INVERTERS (A)	74LS04	11
SINP	NOR GATES (A)	74LS02	5
SINP	HOST CONNECTOR	40P CON.	3
SOUT	NOR GATES (A)	74LS02	6
SOUT	HOST CONNECTOR	40P CON.	4
SYSCLK	INTEL PROCESSOR	8085	37
SYSCLK	DUAL D FLOPS (B)	74LS74	11
SYSCLK	TEST CONNECTOR	32P CON.	33
TX0	"X" OUTPUT PORT	8212	4
TX0	TEST CONNECTOR	32P CON.	10
TX1	"X" OUTPUT PORT	8212	6
TX1	TEST CONNECTOR	32P CON.	25
TX2	"X" OUTPUT PORT	8212	8
TX2	TEST CONNECTOR	32P CON.	9
TX3	"X" OUTPUT PORT	8212	10
TX3	TEST CONNECTOR	32P CON.	26
TX4	"X" OUTPUT PORT	8212	15
TX4	TEST CONNECTOR	32P CON.	8
TX5	"X" OUTPUT PORT	8212	17
TX5	TEST CONNECTOR	32P CON.	27
TX6	"X" OUTPUT PORT	8212	19
TX6	TEST CONNECTOR	32P CON.	7
TX7	"X" OUTPUT PORT	8212	21
TX7	TEST CONNECTOR	32P CON.	28
TY0	"Y" OUTPUT PORT	8212	4
TY0	TEST CONNECTOR	32P CON.	6
TY1	"Y" OUTPUT PORT	8212	6
TY1	TEST CONNECTOR	32P CON.	29
TY2	"Y" OUTPUT PORT	8212	8

TY2	TEST CONNECTOR	32P CON.	5
TY3	"Y" OUTPUT PORT	8212	10
TY3	TEST CONNECTOR	32P CON.	30
TY4	"Y" OUTPUT PORT	8212	15
TY4	TEST CONNECTOR	32P CON.	4
TY5	"Y" OUTPUT PORT	8212	17
TY5	TEST CONNECTOR	32P CON.	31
TY6	"Y" OUTPUT PORT	8212	19
TY6	TEST CONNECTOR	32P CON.	3
TY7	"Y" OUTPUT PORT	8212	21
TY7	TEST CONNECTOR	32P CON.	32
VA00	CRT CONTROLLER	MC6845	4
VA00	NAND GATES (4 INPUT)	74LS20	2
VA01	CRT CONTROLLER	MC6845	5
VA01	NAND GATES (4 INPUT)	74LS20	5
VA02	CRT CONTROLLER	MC6845	6
VA02	SCREEN ADD. SEL. (A)	74LS157	3
VA03	CRT CONTROLLER	MC6845	7
VA03	SCREEN ADD. SEL. (A)	74LS157	6
VA04	CRT CONTROLLER	MC6845	8
VA04	SCREEN ADD. SEL. (A)	74LS157	10
VA05	CRT CONTROLLER	MC6845	9
VA05	SCREEN ADD. SEL. (A)	74LS157	13
VA06	CRT CONTROLLER	MC6845	10
VA06	SCREEN ADD. SEL. (B)	74LS157	3
VA07	CRT CONTROLLER	MC6845	11
VA07	SCREEN ADD. SEL. (B)	74LS157	6
VA08	CRT CONTROLLER	MC6845	12
VA08	SCREEN ADD. SEL. (B)	74LS157	10
VA09	CRT CONTROLLER	MC6845	13
VA09	SCREEN ADD. SEL. (B)	74LS157	13
VA10	CRT CONTROLLER	MC6845	14
VA10	SCREEN ADD. SEL. (C)	74LS157	3
VA11	CRT CONTROLLER	MC6845	15
VA11	SCREEN ADD. SEL. (C)	74LS157	6

VA12	CRT CONTROLLER	MC6845	16
VA12	SCREEN ADD. SEL. (C)	74LS157	10
VA13	CRT CONTROLLER	MC6845	17
VIDCLK2	CRT CONTROLLER	MC6845	21
VIDCLK2	DUAL D FLOP (A)	74LS74	8
VIDCLK2	DUAL D FLOP (A)	74LS74	12
VIDCLK2	NAND GATES (4 INPUT)	74LS20	1
VIDCLK	DUAL D FLOP (A)	74LS74	11
VIDCLK	VIDEO SHIFT REGISTER	74165	2
VIDCLK	DUAL VCO (3 MHZ)	74S124	7
VIDCLK	DUAL D FLOP (C)	74LS74	11
VIDEO	VIDEO SHIFT REGISTER	74165	9
VIDEO	EXCL. OR-NOR GATES	74S86	1
VIDOUT	INVERTERS (C)	74LS14	9
VIDOUT	OR GATES (A)	74LS32	8
VIDO1	EXCL. OR-NOR GATES	74S86	6
VIDO1	EXCL. OR-NOR GATES	74S86	9
VIDO2	EXCL. OR-NOR GATES	74S86	8
VIDO2	DUAL D FLOP (C)	74LS74	12
VIDO3	OR GATES (A)	74LS32	9
VIDO3	DUAL D FLOP (C)	74LS74	9
VSYNC	INVERTERS (C)	74LS14	13
VSYNC	CRT CONTROLLER	MC6845	40
WRSI	NOR GATES (B)	74LS02	4
WRSI	NOR GATES (B)	74LS02	11
WRSI	NOR GATES (B)	74LS02	12
WRS1	INVERTERS (A)	74LS04	1
WRS1	NOR GATES (B)	74LS02	1
WRS1	NOR GATES (B)	74LS02	9
XSEL	INVERTERS (B)	74LS04	4
XSEL	"X" OUTPUT PORT	8212	13
XTAL	INTEL PROCESSOR	8085	1
XTAL	INTEL PROCESSOR	8085	2
YSEL	INVERTERS (B)	74LS04	6
YSEL	"Y" OUTPUT PORT	8212	13
5V	INTEL PROCESSOR	8085	40
5V	UTILITY RAM AND I/O	8155	40
5V	ALDR BUS DEMUX	8212	13

5V	ADDR BUS DEMUX	8212	14
5V	ADDR BUS DEMUX	8212	24
5V	DEVICE SEL DECODER	74LS138	6
5V	DEVICE SEL DECODER	74LS138	16
5V	PROGRAM ROM	2716	21
5V	PROGRAM ROM	2716	24
5V	AND GATES (A)	74LS08	14
5V	INVERTERS (B)	74LS04	14
5V	NOR GATES (A)	74LS02	14
5V	DATA SEL. LATCH (U)	74LS298	16
5V	DATA SEL. LATCH (L)	74LS298	16
5V	BUS DRIVER (B)	74LS365	16
5V	BUS DRIVER (D)	74LS365	16
5V	BUS DRIVER (A)	74LS365	15
5V	BUS DRIVER (C)	74LS365	16
5V	(R) (S) LATCHES	74LS279	16
5V	INVERTERS (A)	74LS04	14
5V	NOR GATES (B)	74LS02	14
5V	"X" OUTPUT PORT	8212	2
5V	"X" OUTPUT PORT	8212	14
5V	"X" OUTPUT PORT	8212	24
5V	"Y" OUTPUT PORT	8212	2
5V	"Y" OUTPUT PORT	8212	14
5V	"Y" OUTPUT PORT	8212	24
5V	DATA BUS DRIVER	74LS245	20
5V	INVERTERS (C)	74LS14	14
5V	OR GATES (C)	74LS32	14
5V	OR GATES (B)	74LS32	14
5V	DUAL D FLOPS (B)	74LS74	2
5V	DUAL D FLOPS (B)	74LS74	4
5V	DUAL D FLOPS (B)	74LS74	10
5V	DUAL D FLOPS (B)	74LS74	13
5V	DUAL D FLOPS (B)	74LS74	14
5V	CHARACTER GEN. (U)	2513	24
5V	CHARACTER GEN. (L)	2513	24
5V	CRT CONTROLLER	MC6845	20
5V	SCREEN BUS DRIVER	74LS245	20
5V	SCREEN ADD. SEL. (A)	74LS157	16
5V	SCREEN ADD. SEL. (B)	74LS157	16
5V	SCREEN ADD. SEL. (C)	74LS157	16
5V	DUAL D FLOP (A)	74LS74	1
5V	DUAL D FLOP (A)	74LS74	4
5V	DUAL D FLOP (A)	74LS74	10
5V	DUAL D FLOP (A)	74LS74	13
5V	DUAL D FLOP (A)	74LS74	14
5V	SCREEN RAM (A)	2114	18
5V	SCREEN RAM (B)	2114	18
5V	SCREEN RAM (C)	2114	18
5V	SCREEN RAM (D)	2114	18
5V	VIDEO SHIFT REGISTER	74165	16
5V	NAND GATES (4 INPUT)	74LS20	4
5V	NAND GATES (4 INPUT)	74LS20	14
5V	DUAL VCO (3 MHZ)	74S124	3

5V	DUAL VCO (3 MHZ)	74S124	15
5V	DUAL VCO (3 MHZ)	74S124	16
5V	ONE SHOT (100 NSEC)	74122	2
5V	ONE SHOT (100 NSEC)	74122	3
5V	ONE SHOT (100 NSEC)	74122	4
5V	ONE SHOT (100 NSEC)	74122	5
5V	ONE SHOT (100 NSEC)	74122	14
5V	EXCL. OR-NOR GATES	74S86	5
5V	EXCL. OR-NOR GATES	74S86	14
5V	3 TO 8 DECODER	74LS138	16
5V	COINCIDENCE GATE (A)	74LS266	14
5V	COINCIDENCE GATE (B)	74LS266	14
5V	OR GATES (A)	74LS32	14
5V	DUAL D FLOP (C)	74LS74	1
5V	DUAL D FLOP (C)	74LS74	4
5V	DUAL D FLOP (C)	74LS74	14
5V	HOST CONNECTOR	40P CON.	39
5V	TEST CONNECTOR	32P CON.	34
5V	PORT ADDR. SELECT	DIP SW.	9
5V	PORT ADDR. SELECT	DIP SW.	10
5V	PORT ADDR. SELECT	DIP SW.	11
5V	PORT ADDR. SELECT	DIP SW.	12
5V	PORT ADDR. SELECT	DIP SW.	13
5V	PORT ADDR. SELECT	DIP SW.	14
5V	PORT ADDR. SELECT	DIP SW.	15
5V	PORT ADDR. SELECT	DIP SW.	16
(ALE)	(R) (S) LATCHES	74LS279	5
(ALE)	INVERTERS (A)	74LS04	4
(ALE)	DUAL D FLOP (A)	74LS74	3
(A06)	INVERTERS (B)	74LS04	10
(A06)	OR GATES (C)	74LS32	5
(BUFFEN)	OR GATES (A)	74LS32	11
(BUFFEN)	HOST CONNECTOR	40P CON.	38
(CERAM)	UTILITY RAM AND I/O	8155	8
(CERAM)	AND GATES (A)	74LS08	3
(CERAM)	INVERTERS (C)	74LS14	1
(CGENS)	DEVICE SEL DECODER	74LS138	9
(CGENS)	OR GATES (B)	74LS32	2
(CGENS)	OR GATES (E)	74LS32	5
(CGWAIT)	AND GATES (A)	74LS08	9
(CGWAIT)	OR GATES (E)	74LS32	3
(CLKI)	INVERTERS (A)	74LS04	5
(CLKI)	NOR GATES (B)	74LS02	13
(CLKV)	INVERTERS (A)	74LS04	13
(CLKV)	NOR GATES (B)	74LS02	10

(COMSTB)	UTILITY RAM AND I/O	8155	39
(COMSTB)	OR GATES (A)	74LS32	3
(CRD)	OR GATES (C)	74LS32	1
(CRD)	OR GATES (C)	74LS32	4
(CRD)	OR GATES (E)	74LS32	6
(CRTS)	DEVICE SEL DECODER	74LS138	7
(CRTS)	OR GATES (C)	74LS32	10
(CRTS)	CRT CONTROLLER	MC6845	25
(CRWAIT)	AND GATES (A)	74LS08	10
(CRWAIT)	OR GATES (C)	74LS32	8
(CSTBE)	3 TO 8 DECODER	74LS138	13
(CSTBE)	OR GATES (A)	74LS32	2
(DATASTB)	UTILITY RAM AND I/O	8155	5
(DATASTB)	OR GATES (A)	74LS32	6
(DSTBE)	3 TO 8 DECODER	74LS138	14
(DSTBE)	OR GATES (A)	74LS32	5
(HINOUT)	NOR GATES (A)	74LS02	4
(HINOUT)	3 TO 8 DECODER	74LS138	5
(HRD)	INVERTERS (B)	74LS04	8
(HRD)	BUS DRIVER (A)	74LS365	15
(HRD)	BUS DRIVER (C)	74LS365	15
(HRD)	OR GATES (A)	74LS32	12
(HSEL)	BUS DRIVER (A)	74LS365	1
(HSEL)	BUS DRIVER (C)	74LS365	1
(HSEL)	NOR GATES (B)	74LS02	6
(HSEL)	3 TO 8 DECODER	74LS138	15
(HSEL)	OR GATES (A)	74LS32	13
(HSYNC)	INVERTERS (C)	74LS14	10
(HWR)	NOR GATES (B)	74LS02	5
(HWR)	OR GATES (A)	74LS32	1
(HWR)	OR GATES (A)	74LS32	4
(HWR)	HOST CONNECTOR	40P CON.	2
(LCEN)	OR GATES (C)	74LS32	3
(LCEN)	CHARACTER GEN. (L)	2513	11
(LOAD)	NAND GATES (4 INPUT)	74LS20	6
(LOAD)	ONE SHOT (100 NSEC)	74122	1
(LOAD)	DUAL D FLOP (C)	74LS74	3
(PLOAD)	VIDEO SHIFT REGISTER	74165	1

(FLOAD)	ONE SHOT (100 NSEC)	74122	6
(PORTS)	DEVICE SEL DECODER	74LS138	10
(PORTS)	AND GATES (A)	74LS08	1
(PORTS)	INVERTERS (B)	74LS04	1
(RAMS)	DEVICE SEL DECODER	74LS138	14
(RAMS)	AND GATES (A)	74LS08	2
(RDWR)	AND GATES (A)	74LS08	6
(RDWR)	INVERTERS (C)	74LS14	5
(RDWR)	OR GATES (B)	74LS32	10
(RDY)	OR GATES (C)	74LS32	9
(RDY)	OR GATES (B)	74LS32	1
(RDY)	DUAL D FLOPS (B)	74LS74	1
(RDY)	DUAL D FLOPS (B)	74LS74	8
(RD)	INTEL PROCESSOR	8085	32
(RD)	UTILITY RAM AND I/O	8155	9
(RD)	AND GATES (A)	74LS08	4
(RD)	NOR GATES (A)	74LS02	2
(RD)	BUS DRIVER (B)	74LS365	1
(RD)	BUS DRIVER (D)	74LS365	1
(RD)	OR GATES (B)	74LS32	4
(ROMS1)	DEVICE SEL DECODER	74LS138	15
(ROMS1)	NOR GATES (A)	74LS02	3
(ROMS)	PROGRAM ROM	2716	20
(ROMS)	INVERTERS (B)	74LS04	12
(RST-IN)	INTEL PROCESSOR	8085	36
(RST-IN)	CRT CONTROLLER	MC6845	2
(RST-IN)	TEST CONNECTOR	32P CON.	24
(SA10)	INVERTERS (C)	74LS14	4
(SA10)	SCREEN RAM (B)	2114	8
(SA10)	SCREEN RAM (D)	2114	8
(SELSC)	OR GATES (B)	74LS32	13
(SELSC)	SCREEN BUS DRIVER	74LS245	19
(SELSC)	SCREEN ADL. SEL. (A)	74LS157	1
(SELSC)	SCREEN ADL. SEL. (B)	74LS157	1
(SELSC)	SCREEN ADL. SEL. (C)	74LS157	1
(SELSC)	DUAL D FLOP (A)	74LS74	6
(SEL)	DATA SEL. LATCH (U)	74LS298	10
(SEL)	DATA SEL. LATCH (L)	74LS298	10
(SEL)	INVERTERS (A)	74LS04	10
(STATS)	DEVICE SEL DECODER	74LS138	11
(STATS)	BUS DRIVER (B)	74LS365	15

(STATS)	BUS DRIVER (D)	74LS365	15
(STATS)	NOR GATES (B)	74LS02	2
(UCEN)	OR GATES (C)	74LS32	6
(UCEN)	CHARACTER GEN. (U)	2513	11
(VIDEN)	OR GATES (A)	74LS32	10
(VIDEN)	DUAL D FLOP (C)	74LS74	6
(VIDOUT)	INVERTERS (C)	74LS14	8
(VSYNC)	INVERTERS (C)	74LS14	12
(WRS1)	(R) (S) LATCHES	74LS279	6
(WRS1)	INVERTERS (A)	74LS04	2
(WR)	INTEL PROCESSOR	8085	31
(WR)	UTILITY RAM AND I/O	8155	10
(WR)	AND GATES (A)	74LS08	5
(WR)	NOR GATES (B)	74LS02	3
(WR)	"X" OUTPUT PORT	8212	1
(WR)	"Y" OUTPUT PORT	8212	1
(WR)	DATA BUS DRIVER	74LS245	1
(WR)	OR GATES (B)	74LS32	12
(XSEL)	DEVICE SEL DECODER	74LS138	12
(XSEL)	INVERTERS (B)	74LS04	3
(YSEL)	DEVICE SEL DECODER	74LS138	13
(YSEL)	INVERTERS (B)	74LS04	5

THE FOLLOWING LABELS ARE USED ONLY ONCE.

C27FF
HDI4
HDI5
HDI6
HDI7
PCT
RCEXT
VA13
(HSYNC)
(VIDOUT)
(VSYNC)

********** PARTS LIST **********

PART NUMBER	QUANTITY
DIP SW.	1
MC6845	1
2114	4
2513	2
2716	1
32P CON.	1
40P CON.	1
74LS02	2
74LS04	2
74LS08	1
74LS138	2
74LS14	1
74LS157	3
74LS20	1
74LS245	2
74LS266	2
74LS279	1
74LS298	2
74LS32	3
74LS365	4
74LS74	3
74S124	1
74S86	1
74122	1
74165	1
8085	1
8155	1
8212	3

A.7 Source Listing of the Wirelist Program

```
PROGRAM WIRELIST(INPUT,OUTPUT);
(*****************************************************************)
(*  WIRELIST- THE PURPOSE OF THIS PROGRAM IS TO PRODUCE A        *)
(*  LISTING OF ALL THE WIRES USED IN THE REALIZATION OF A        *)
(*  DIGITAL DESIGN.                                              *)
(*                                                               *)
(*  INPUT- A CHIP-BY-CHIP LISTING OF THE DEVICES USED. EACH      *)
(*         CHIP IS DESCRIBED BY THE FOLLOWING (TYPED ONE TO      *)
(*         A LINE):                                              *)
(*                1) CHIP TYPE (UP TO 20 CHARACTERS)             *)
(*                2) CHIP NUMBER (UP TO 10 CHARACTERS)           *)
(*                3) NUMBER OF PINS (INTEGER)                    *)
(*              4-?) THE NAME OF THE WIRE CONNECTED TO EACH      *)
(*                   PIN (UP TO 10 CHARACTERS). INVERTED         *)
(*                   SIGNALS ARE ENCLOSED IN PARENTHESIS.        *)
(*                   THE END OF THE DATA IS SIGNALED BY A        *)
(*                   CARD CONTAINING THE WORD 'DONE'.            *)
(*                                                               *)
(*  OUTPUT- THE PROGRAM PRODUCES THREE SEPARATE LISTINGS:        *)
(*          1) A LIST SHOWING EACH CHIP WITH ALL OF THE WIRES    *)
(*             WHICH CONNECT TO IT.                              *)
(*          2) A CROSS REFERENCE LISTING. THIS IS AN ALPHA-      *)
(*             BETIZED LIST SHOWING THE POINTS TO WHICH EACH     *)
(*             WIRE IS CONNECTED.                                *)
(*          3) A PARTS LIST GIVING THE QUANTITIES OF EACH        *)
(*             CHIP USED IN THE DESIGN.                          *)
(*                                                            *)
(*  NOTES-                                                       *)
(*          1) WIRES LABELED 'NC' ARE NOT LISTED IN THE CROSS    *)
(*             REFERENCE.                                        *)
(*          2) DURING THE CONTRUCTION OF A PROJECT IT MAY BE     *)
(*             USEFUL TO DISTINGUISH THOSE WIRES WHICH ARE       *)
(*             NOT YET CONNECTED. THIS MAY BE DONE BY PLACING    *)
(*             A '*' AFTER A WIRE LABEL. THIS ASTERISK IS        *)
(*             IGNORED BY THE CROSS REFERENCER.               *)
(*                                                               *)
(*  AUTHOR- ERIC MCKINLAY                                        *)
(*  DATE- SPRING 1979                                            *)
(*                                                               *)
(*****************************************************************)
CONST CHIPCOUNT   = 100;
      PAGELEN     = 60;
      MAXPINS     = 40;
      MAXLABS     = 400;
      LABELLENGTH = 10;
      NAMELENGTH  = 20;
      BLANKS      = '          ';
      NCONNECTED  = 'NC        ';
```

```
      LBLANKS      = '                    ';
      DONE         = 'DONE                ';
      MOVE         = -3;
      DRAW         = -2;

TYPE LINELABEL = PACKED ARRAY[1..LABELLENGTH] OF CHAR;
     NAMELABEL = PACKED ARRAY[1..NAMELENGTH] OF CHAR;
     SOCKET    = RECORD
                    NAME: NAMELABEL;
                    CHIPTYPE: LINELABEL;
                    PINS: INTEGER;
                    CONNECT: ARRAY[1..MAXPINS] OF LINELABEL;
                 END;

VAR CHIP: ARRAY[1..CHIPCOUNT] OF SOCKET;
    NCHIP,MAXX,MAXY: INTEGER;
    LABLIST, SINGLELABS: ARRAY[1..MAXLABS] OF LINELABEL;
    NLABS, SINGLECOUNT: INTEGER;
    LINESPRINTED: INTEGER;
    PARTS: ARRAY[1..CHIPCOUNT] OF LINELABEL;


  (************************ INITIALIZATION ************************)
  PROCEDURE INITIALIZE;
  VAR I: INTEGER;
  BEGIN
     LINESPRINTED := 0;
     WRITE('1','********************************* WIRELIST -');
     WRITELN(' VERSION 1.0 *********************************');
     LINELIMIT(OUTPUT,5000);
     SINGLECOUNT := 0;
  END;


(*****************************************************************)
(* IF THE WIRE LABEL ENDS WITH A STAR (INDICATING THAT THE       *)
(* WIRE IS NOT YET CONNECTED) DELETE THE STAR BEFORE INSERTING   *)
(* THE LABEL IN THE LIST OF DISTINCT LABELS.                     *)
(*****************************************************************)
  PROCEDURE DELETESTARS(VAR TEXTSTRING: LINELABEL);
  VAR I: INTEGER;
      USTRING: ARRAY[1..LABELLENGTH] OF CHAR;
  BEGIN
     IF TEXTSTRING <> BLANKS THEN
     BEGIN
        UNPACK(TEXTSTRING,USTRING,1);
        I := LABELLENGTH;
        WHILE USTRING[I] = ' ' DO I := I-1;
        IF USTRING[I] = '*' THEN
        BEGIN
           USTRING[I] := ' ';
           PACK(USTRING,1,TEXTSTRING);
        END;
```

```
    END;
  END;


(*****************************************************************)
(* ADD A WIRE LABEL TO THE LIST OF DISTINCT LABELS IF IT IS      *)
(* NOT ALREADY IN THE LIST.                                      *)
(*****************************************************************)
  PROCEDURE ADDNAME(LABELNAME: LINELABEL);
  VAR I: INTEGER;
  BEGIN
    IF (LABELNAME <> BLANKS) AND (LABELNAME <> NCONNECTED) THEN
    BEGIN
      DELETESTARS(LABELNAME);
      IF NLABS <> 0 THEN
      BEGIN
        I := 1;
        WHILE (I<NLABS) AND (LABLIST[I]<>LABELNAME) DO I := I+1;
        IF LABLIST[I] <> LABELNAME THEN
        BEGIN
          NLABS := NLABS+1;
          LABLIST[NLABS] := LABELNAME;
        END;
      END
      ELSE
      BEGIN
        NLABS := 1;
        LABLIST[NLABS] := LABELNAME;
      END;
    END;
  END;

  FUNCTION MAX(A,B: INTEGER): INTEGER;
  BEGIN
    IF A>B THEN
      MAX := A
    ELSE
      MAX := B;
  END;


(*****************************************************************)
(* READ IN A TEXT STRING OF MAXIMUM LENGTH 10.                   *)
(*****************************************************************)
  PROCEDURE READSHORT(VAR PARRAY: LINELABEL);
  VAR I: INTEGER;
      CARRAY: ARRAY[1..LABELLENGTH] OF CHAR;
  BEGIN
    I := 0;
    WHILE (I<LABELLENGTH) AND (NOT EOLN) DO
    BEGIN
      I:=I+1;
      READ(CARRAY[I]);
```

```
    END;
    FOR I:=I+1 TO LABELLENGTH DO CARRAY[I] := ' ';
    PACK(CARRAY,1,PARRAY);
    READLN;
  END;


(*****************************************************************)
(* READ IN A TEXT STRING OF MAXIMUM LENGTH 20.                   *)
(*****************************************************************)
  PROCEDURE READLONG(VAR PARRAY: NAMELABEL);
  VAR I: INTEGER;
      CARRAY: ARRAY[1..NAMELENGTH] OF CHAR;
  BEGIN
    I := 0;
    WHILE (I<NAMELENGTH) AND (NOT EOLN) DO
    BEGIN
      I:=I+1;
      READ(CARRAY[I]);
    END;
    FOR I:=I+1 TO NAMELENGTH DO CARRAY[I] := ' ';
    PACK(CARRAY,1,PARRAY);
    READLN;
  END;


(*****************************************************************)
(* READ IN ALL THE SOCKETS AND MAINTAIN THE PARTS LIST.          *)
(*****************************************************************)
  PROCEDURE READSOCKETS;
  VAR I,J: INTEGER;
      TNAME: NAMELABEL;
      UCHIPTYPE,UCONNECT: ARRAY[1..LABELLENGTH] OF CHAR;
  BEGIN
    NCHIP := 0;
    NLABS := 0;
    READLONG(TNAME);
    WHILE TNAME <> DONE DO
    BEGIN
      NCHIP := NCHIP+1;
      WITH CHIP[NCHIP] DO
      BEGIN
        NAME := TNAME;
        READSHORT(CHIPTYPE);
        PARTS[NCHIP] := CHIPTYPE;
        READLN(PINS);
        FOR I:=1 TO PINS DO
        BEGIN
          READSHORT(CONNECT[I]);
          ADDNAME(CONNECT[I]);
        END;
        FOR I:=PINS+1 TO MAXPINS DO CONNECT[I] := BLANKS;
```

```
      END;
      READLONG(TNAME);
    END;
    WRITELN(' ','THE NUMBER OF CHIPS IS',NCHIP:4,'.');
    WRITELN(' ','THE NUMBER OF DISTINCT LABELS IS',NLABS:4,'.');
  END;


(*************************************************************)
(* PRINT OUT THE LIST OF SOCKETS IN TWO COLUMNS. IF THE TWO  *)
(* SOCKETS WHICH ARE PRINTED SIDE-BY-SIDE HAVE AN UNEQUAL    *)
(* NUMBER OF PINS, THE SHORTER ONE IS PADDED WITH BLANK      *)
(* LABELS.                                                   *)
(*************************************************************)
  PROCEDURE LISTSOCKETS;
  VAR I,SUB1,SUB2,J,PIN1,PIN2,LOC: INTEGER;
  BEGIN
    WRITELN;WRITELN;
    FOR I:=1 TO NCHIP DIV 2 DO
    BEGIN
      SUB2 := 2*I;
      SUB1 := SUB2-1;
      PIN1 := CHIP[SUB1].PINS DIV 2;
      PIN2 := CHIP[SUB2].PINS DIV 2;
      IF (LINESPRINTED+8+MAX(PIN1,PIN2)) > PAGELEN THEN
      BEGIN
        LINESPRINTED := 0;
        WRITELN('1');
      END;
      WRITELN;WRITELN;
      WRITELN(' ','------------------------------',' ':10,
              '------------------------------');
      WRITELN(' ',CHIP[SUB1].NAME,' ':20,CHIP[SUB2].NAME);
      WRITELN(' ',CHIP[SUB1].CHIPTYPE,' ':30,CHIP[SUB2].CHIPTYPE);
      WRITELN(' ',CHIP[SUB1].PINS:2,' PIN',' ':34,
              CHIP[SUB2].PINS:2,' PIN');
      WRITELN(' ','------------------------------',' ':10,
              '------------------------------');
      FOR J:=1 TO MAX(PIN1,PIN2) DO
      BEGIN
        IF J<=PIN1 THEN
        BEGIN
          LOC := PIN1*2 - (J-1);
          WRITE(' ',J:2,' ',CHIP[SUB1].CONNECT[J]);
          WRITE(' ':4);
          WRITE(LOC:2,' ',CHIP[SUB1].CONNECT[LOC]);
          WRITE(' ':10);
        END
        ELSE
          WRITE(' ':40);
        IF J <= PIN2 THEN
        BEGIN
          LOC := PIN2*2 - (J-1);
```

```
            WRITE(' ',J:2,' ',CHIP[SUB2].CONNECT[J]);
            WRITE(' ':4);
            WRITE(LOC:2,' ',CHIP[SUB2].CONNECT[LOC]);
         END;
         WRITELN;
      END;
      LINESPRINTED := LINESPRINTED + MAX(PIN1,PIN2) + 8;
   END;
END;


(*****************************************************************)
(* PRODUCE AND ALPHABETIZED CROSS REFERENCE LISTING OF ALL OF    *)
(* THE WIRE LABELS.                                              *)
(*****************************************************************)
  PROCEDURE LISTXREF;
  VAR LABELNAME,TEMPSTRING: LINELABEL;
      ILAB,ICHIP,IPIN,OCCURS: INTEGER;


    PROCEDURE SORTLABELS;
    VAR I,J,SM: INTEGER;
        TEMP: LINELABEL;
    BEGIN
      FOR I:=1 TO (NLABS-1) DO
      BEGIN
        SM := I;
        FOR J:=I+1 TO NLABS DO
          IF LABLIST[J] < LABLIST[SM] THEN SM := J;
        TEMP := LABLIST[I];
        LABLIST[I] := LABLIST[SM];
        LABLIST[SM] := TEMP;
      END;
    END;


  BEGIN
    WRITELN;WRITELN;WRITELN;WRITELN;
    WRITELN('1','************************ CROSS REFERENCE',
            ' LISTING ************************');
    WRITELN;WRITELN;
    SORTLABELS;
    FOR ILAB:=1 TO NLABS DO
    BEGIN
      LABELNAME := LABLIST[ILAB];
      OCCURS := 0;
      WRITELN;WRITELN;
      FOR ICHIP:=1 TO NCHIP DO
        FOR IPIN:=1 TO CHIP[ICHIP].PINS DO
        BEGIN
          TEMPSTRING := CHIP[ICHIP].CONNECT[IPIN];
          DELETESTARS(TEMPSTRING);
          IF LABELNAME = TEMPSTRING THEN
```

```
            BEGIN
               WRITELN(' ',CHIP[ICHIP].CONNECT[IPIN],CHIP[ICHIP].
                   NAME,' ':5,CHIP[ICHIP].CHIPTYPE,IPIN:7);
               OCCURS := OCCURS+1;
            END;
         END;
      IF OCCURS = 1 THEN
      BEGIN
         SINGLECOUNT := SINGLECOUNT+1;
         SINGLELABS[SINGLECOUNT] := LABELNAME;
      END;
   END;
END;


(*************************************************************)
(* PRINT A LIST OF THOSE LABELS WHICH ARE USED ONLY ONCE.    *)
(*************************************************************)
  PROCEDURE PRINTSINGLES;
  VAR I: INTEGER;
  BEGIN
     WRITELN;WRITELN;WRITELN;WRITELN;
     IF SINGLECOUNT <> 0 THEN
     BEGIN
        WRITELN('1','THE FOLLOWING LABELS ARE USED ONLY ONCE.');
        WRITELN(' ','-------------------------------------');
        WRITELN;
        FOR I:=1 TO SINGLECOUNT DO
           WRITELN(' ',SINGLELABS[I]:20);
     END
     ELSE
        WRITELN('1','ALL LABELS WERE USED AT LEAST ONCE.');
  END;


(*************************************************************)
(* PRINT A LIST OF THE PARTS USED IN THE DESIGN.             *)
(*************************************************************)
  PROCEDURE PRINTPARTSLIST;
  VAR TEMP, OLDIY: LINELABEL;
      I, EACH, SM, J: INTEGER;
  BEGIN
        FOR I:=1 TO (NCHIP-1) DO
        BEGIN
           SM := I;
           FOR J:=I+1 TO NCHIP DO
              IF PARTS[J] < PARTS[SM] THEN SM := J;
           TEMP := PARTS[I];
           PARTS[I] := PARTS[SM];
           PARTS[SM] := TEMP;
        END;
        WRITE('1','****************************** PARTS LIST');
        WRITELN('******************************');
```

```
      WRITELN;WRITELN;WRITELN;
      WRITELN('  ','PART NUMBER          QUANTITY');
      WRITELN('  ','-----------          --------');
    EACH := 1;
    OLDTY := BLANKS;
    PARTS[NCHIP+1] := BLANKS; (* SENTINEL *)
    FOR I:=1 TO NCHIP+1 DO
      IF(OLDTY = PARTS[I]) THEN
        EACH := EACH+1
      ELSE
      BEGIN
        IF(OLDTY <> BLANKS) THEN WRITELN('  ',OLDTY:10,EACH:13);
        EACH := 1;
        OLDTY := PARTS[I];
      END;
  END;


BEGIN (* TINY MAIN PROGRAM *)
  INITIALIZE;
  READSOCKETS;
  LISTSOCKETS;
  LISTXREF;
  PRINTSINGLES;
  PRINTPARTSLIST;
END.
```

A.8 Suggestions for Enhancements

The following are suggestions for future improvements to the MPIX System. It is believed by the author that the system provides a solid basis for a more complete graphics system.

1) Interactive Capabilities. The current system was designed to act only as an output device. However, with fairly minor changes, the system can provide a moderate ammount of interaction. In particular the light pen register within the CRTC (currently unused) makes addition of a pointing device, such as a light pen or cross hairs, fairly simple. The only major hardware change would be the addition of one or more output ports to send information to the host.

2) Display Files. The current system is only capable of handling a single vector at a time. A more useful system would allow the user to define graphical objects. This could be implemented with a considerably more complex program resident on the MPIX, requiring the use of more ROM and restructuring the MPIX address space.

3) Status Port. At the time the MPIX was designed the author could find no single chip which could be used for the status port. It therefore was constructed using 6 separate chips. It is likely that with currently available chips, the task of the status port could be done with a single device. The author believes that the 74172 (dual port memory) would be a reasonable choice for this register.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCSCS-R-79-971	2.	3. Recipient's Accession No.

4. Title and Subtitle

MPIX: A MICROPROCESSOR-BASED SYSTEM FOR GRAPHICAL OUTPUT

5. Report Date: June, 1979

6.

7. Author(s): Eric Wallace McKinlay

8. Performing Organization Rept. No.

9. Performing Organization Name and Address

Department of Computer Science
University of Illinois
Urbana, IL 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

13. Type of Report & Period Covered: Thesis

14.

15. Supplementary Notes

16. Abstracts

This paper contains a complete description of the design, implementation and realization of a microprocessor graphics processor. The processor (called MPIX) operates in parallel with a host processor and produces output on a television receiver. The paper contains complete listings of all software for the MPIX as well as descriptions of several novel tools used in the system design.

17. Key Words and Document Analysis. 17a. Descriptors

Computer graphics, microprocessor, video, system design

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 123
	20. Security Class (This Page UNCLASSIFIED	22. Price

FEB 20 1991

UNIVERSITY OF ILLINOIS-URBANA
3 0112 045816599